Source file: nfetc

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
############################################################

############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
############################################################

############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?        'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
############################################################

############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
#############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
#############################################################

#############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
#############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

#############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
#############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

#############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
#############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

#############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
#############################################################
einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

#############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
#############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

#############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
#############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

#############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
############################################################

############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: `nfetc`

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```

Source file: nfetc

```
##############################################################
# Illustrates use of some utility commands available
# on einstein (although 'paste' is a generic Unix command)
# which are useful for manipulating columns of numbers.
#
# (1) dvmesh: Generates uniform sequence of real numbers.
#     Included here mostly as a mechanism to generate
#     input for 'nf'.  (Instructor-supplied C-program).
#
# (2) nf: Generalization of 'nth'. (See Course Notes for
#     September 25th.)  Filter which selects columns from
#     standard input (assumed numeric), performs fairly
#     general mathematical operations as needed, and outputs
#     one or more columns of numbers on standard output.
#     (Instructor supplied perl-script).
#
# (3) paste: Standard Unix facility for combining ('pasting')
#     one or more file arguments.,
##############################################################

##############################################################
# Usage of 'dvmesh' is straightforward.  The command
# generates a length 'n' sequence of real numbers, uniformly
# spaced, and ranging from 'xmin' to 'xmax'.
##############################################################

einstein% dvmesh
usage: dvmesh <xmin> <xmax> <n > 0>

einstein% dvmesh 0.0 1.0 11
  0.0000000000000000E+00
  1.0000000000000001E-01
  2.0000000000000001E-01
  3.0000000000000004E-01
  4.0000000000000002E-01
  5.0000000000000000E-01
  5.9999999999999998E-01
  6.9999999999999996E-01
  7.9999999999999993E-01
  8.9999999999999991E-01
  1.0000000000000000E+00

##############################################################
# 'nf' accepts an arbitrary number of arguments, reads
# columns of numbers from standard input, then manipulates
# the input-columns and writes the results to standard
# output.  Use the notation '_1', '_2' etc. to refer to
# the first, second etc. column.  Usage is best demonstrated
# with some examples:
##############################################################

einstein% nf
usage: nf <expr #> [<expr #> ...]

##############################################################
# Compute x^2, x = 0.0, 0.1, ... 0.9,  1.0 and write
# (x, x^2) to standard output.  Note use of single quotes
# around 2nd argument to 'nf' to inhibit shell-interpretation
# of multiplication operator '*'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1'
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

##############################################################
# Repeat the calculation and redirect to a file 'squares'.
# Compute the cubes of the same x-values and redirect
# (x,x^3) to 'cubes'.
##############################################################

einstein% dvmesh 0.0 1.0 11 | nf _1 '_1 * _1' > squares
einstein% cat squares
0.0000000000000000E+00   0
1.0000000000000001E-01   0.01
2.0000000000000001E-01   0.04
3.0000000000000004E-01   0.09
4.0000000000000002E-01   0.16
5.0000000000000000E-01   0.25
5.9999999999999998E-01   0.36
6.9999999999999996E-01   0.49
7.9999999999999993E-01   0.64
8.9999999999999991E-01   0.81
1.0000000000000000E+00   1

einstein% dvmesh 0.0 1.0 11 | nf _1 'pow(_1,3)' > cubes
einstein% cat cubes
0.0000000000000000E+00   0
1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.729
1.0000000000000000E+00   1

##############################################################
# 'nf' understands
#
# (A) The usual binary arithmetic operations: *, /, +, -,
# (B) Integer power function (uses repeated multiplies)
#     ipow(ix,iy) = ix^iy
# (C) Real power function (uses logs and exponentiation)
#     pow(x,y) = x^y (x must be postive-definite)
# (D) min() and max() of an arbitrary number of arguments
# (E) The usual suite of mathematical functions: sin, cos,
#     tan, sinh, cosh, tanh, exp, log, abs, sqrt (inverse
#     trig and hyperbolic function are currently *not*
#     implemented.
##############################################################

einstein% dvmesh 0.0 4.0 11 | nf _1 'sin(_1)' 'cos(_1)' \
?         'ipow(sin(_1),2) + ipow(cos(_1),2)'
0.0000000000000000E+00   0   1   1
4.0000000000000002E-01   0.389418342308651   0.921060994002885   1
8.0000000000000004E-01   0.717356090899523   0.696706709347165   1
1.2000000000000002E+00   0.932039085967226   0.362357754476673   1
1.6000000000000001E+00   0.999573603041505   -0.0291995223012889   1
2.0000000000000000E+00   0.909297426825682   -0.416146836547142   1
2.3999999999999999E+00   0.675463180551151   -0.737393715541246   1
2.7999999999999998E+00   0.334988150155905   -0.942222340668658   1
3.1999999999999997E+00   -0.0583741434275798   -0.998294775794753   1
3.5999999999999996E+00   -0.442520443294852   -0.896758416334147   1
4.0000000000000000E+00   -0.756802495307928   -0.653643620863612   1

##############################################################
# 'paste': Combines files 'horizontally' in a straightforward
# fashion.  Most useful for use with two or more files each
# of which contain one or more columns with, but
# which all contain the same number of lines (length of
# columns).  Note that paste's output is to standard out.
##############################################################

einstein% paste squares cubes
0.0000000000000000E+00   0      0.0000000000000000E+00   0
1.0000000000000001E-01   0.01   1.0000000000000001E-01   0.001
2.0000000000000001E-01   0.04   2.0000000000000001E-01   0.008
3.0000000000000004E-01   0.09   3.0000000000000004E-01   0.027
4.0000000000000002E-01   0.16   4.0000000000000002E-01   0.064
5.0000000000000000E-01   0.25   5.0000000000000000E-01   0.125
5.9999999999999998E-01   0.36   5.9999999999999998E-01   0.216
6.9999999999999996E-01   0.49   6.9999999999999996E-01   0.343
7.9999999999999993E-01   0.64   7.9999999999999993E-01   0.512
8.9999999999999991E-01   0.81   8.9999999999999991E-01   0.729
1.0000000000000000E+00   1      1.0000000000000000E+00   1

##############################################################
# The above is probably not quite what we wanted.  Use
```

```
# 'nf' (or 'nth') to get rid of third column.  Note that
# 'nth' refers to columnm 1, 2 etc simply as '1', '2'.
############################################################

einstein% paste squares cubes | nf _1 _2 _4
0.0000000000000000E+00   0   0
1.0000000000000001E-01   0.01   0.001
2.0000000000000001E-01   0.04   0.008
3.0000000000000004E-01   0.09   0.027
4.0000000000000002E-01   0.16   0.064
5.0000000000000000E-01   0.25   0.125
5.9999999999999998E-01   0.36   0.216
6.9999999999999996E-01   0.49   0.343
7.9999999999999993E-01   0.64   0.512
8.9999999999999991E-01   0.81   0.729
1.0000000000000000E+00   1   1

einstein% paste squares cubes | nth 1 2 4
0.0000000000000000E+00 0 0
1.0000000000000001E-01 0.01 0.001
2.0000000000000001E-01 0.04 0.008
3.0000000000000004E-01 0.09 0.027
4.0000000000000002E-01 0.16 0.064
5.0000000000000000E-01 0.25 0.125
5.9999999999999998E-01 0.36 0.216
6.9999999999999996E-01 0.49 0.343
7.9999999999999993E-01 0.64 0.512
8.9999999999999991E-01 0.81 0.729
1.0000000000000000E+00 1 1
```