PHY/CAM 381C: Computational Physics   Homework 2
Due: Thursday, February 20, 12:30pm

Important: *The first four questions of this assignment require working with the* `xmaple` *graphical interface to produce* Maple *worksheets as well as preparing source code for* Maple *procedures in plain-text files which can be input into* `maple` *or* `xmaple` *via the* `read` *command. You can run* `xmaple` *and* `maple` *on either the SGI machines or the Linux machines. However, in order to complete the problem set, all files required by the assignment must reside in your account on the SGIs. If you prepare and save your worksheets, source files etc. on the Linux machines, be sure to copy them to the correct places in your SGI account using* `rcp` *or* `ftp`*. As always, let me know immediately if there is something which you don't understand, or if you encounter serious problems with any part of the assignment.*

Problem 1: Using Chapter 2 of the *Maple V Learning Guide*, make and save a *Maple* worksheet like the one I went through in class on January 28. (Note that a Postscript version of my worksheet is available on line via the *Class Notes* web page.) You are to work though Chapter 2 *in its entirety*, entering everything which follows a *Maple* prompt (`>`) into your worksheet. Your worksheet should also contain annotations (descriptive text added with the `Insert Text` function of the `Edit` menu) corresponding to the various sections and sub-sections of the Chapter, as mine does. Save the worksheet in the file `~/hw2/a1/the.ms` on your SGI account. Note that `.ms` is the standard extension for `xmaple` worksheet files. You need to do this question using the GUI (i.e. using `xmaple`) and you are warned that it may take *a few hours* to complete. Note that any plots which are generated by your worksheet will not appear *in* the worksheet: this is normal and you should not be concerned by it.

Problem 2: Create a worksheet called `~/hw2/a2/the.ms` in which the following computations have been performed:

$$\frac{d^3}{dx^3}\left(\sin\left(x\right)\cos\left(x\right)+\exp\left(\tanh\left(1/x\right)\right)\right) \tag{1.1}$$

$$\int \frac{x^3+4x^2+6}{x^2-1}\,dx \tag{1.2}$$

$$\int_{y=-3}^{y=3}\int_{x=-2}^{x=2}\frac{x^2+2xy+y^2}{x^2-y^2}\,dxdy \tag{1.3}$$

$$\text{Taylor series about } x=0,\ \text{up to and including the } O(x^{10}) \text{ term, of } \exp(\sin(x)+\cos(3x)) \tag{1.4}$$

Problem 3: Write a *Maple* procedure called `reverse` which (a) reverses the order of elements of a list, and (b) reverses the order of characters in a string. For any other type of input, the procedure should print a suitable error message and exit. Here are some sample invocations and the resulting output:

```
> # Input definitions from file `reverse`
> read reverse;

> reverse([1,2,3,4]);

       [4, 3, 2, 1]

> reverse([]);

       [ ]

> reverse(`This is a string`);

       gnirts a si sihT
```

```
> reverse(2*s);

        reverse: Can not reverse '2*s'
```

You may find it convenient to define one or more auxiliary procedures which are invoked by `reverse`. All source code must be prepared in the text file (i.e. *not* a worksheet) `~/hw2/a3/reverse`. Test your procedure thoroughly on input—both valid and invalid—of your own choosing; I will evaluate your implementation using my own input.

Problem 4: From the following 3 dimension-full physical constants (values given in *SI* units):

- *Newton's gravitational constant:* $G = 6.673 \times 10^{-11}\,\mathrm{kg}^{-1}\mathrm{m}^3\mathrm{s}^{-2}$
- *Speed of light:* $c = 2.998 \times 10^8\,\mathrm{m\,s}^{-1}$
- *Planck's reduced constant:* $\hbar = 1.0546 \times 10^{-34}\,\mathrm{kg\,m}^2\,\mathrm{s}^{-1}$

it is possible to compute a fundamental mass, length, time, density etc. known as the Planck mass, Planck length, Planck time, Planck density etc. More precisely, for any physical attribute with dimension

$$M^{\alpha_1}\, L^{\alpha_2}\, T^{\alpha_3} \tag{3.1}$$

where M, L, and T have the dimensions of mass, length, and time respectively, and the α_i are real constants, the associated Planck quantity has the same dimensions, and is generically given by

$$c^{\beta_1}\, \hbar^{\beta_2}\, G^{\beta_3} \tag{3.2}$$

for some to-be-determined real constants β_i. For example:

$$L \sim c^{-3/2}\, \hbar^{1/2}\, G^{1/2}$$

where the $\sim$ denotes "has the same dimensions". In *SI* units, then, the Planck length is 1.616×10^{-34}m.

Write a *Maple* procedure called `planck` which accepts algebraic expressions of the form (3.1) and returns the corresponding Planck quantity (3.2). You should first extend the `type` procedure to recognize a new type `MLTdim` which is any expression precisely of the form (3.1) (with *constant* α_i). Thus, for example,

```
> type(M,MLTdim);

            true

> type(M^2/(L*T),MLTdim);

            true

> type(2*L,MLTdim);

            false

> type(M^p1,MLTdim);

            false
```

Once you have extended `type` appropriately, note that you can use *Maple*'s type-checking facility by using a header of the form:

```
  planck := proc(x:MLTdim)
```

Finally, extend the floating-point evaluation routine, `evalf` so that it recognizes the constants `G`, `c` and `hbar` and returns their *SI* values (without dimensions) as given above. Prepare all of the procedures you write in a single file called `~/hw2/a4/planck`. Typical output from `planck` should look like this:

```
> # Input definitions from file 'planck'
> read planck;

> planck(L);
                          1/2  1/2
                      hbar    G
                      ------------
                           3/2
                          c
> evalf(");
                                 -34
                   .1616058223*10
```

Test your implementation thoroughly; I will evaluate it using my own input.

Problem 5: In directory `~/hw2/a5`, write a Fortran program `newt3` (source code in `newt3.f`), which finds a root of the following system using Newton's method in 3-dimensions:

$$\begin{aligned} x^2 + y^3 + z^4 &= 1 \\ \sin(xyz) &= x + y + z \\ \cos(x) &= yz \end{aligned}$$

`newt3` should accept 3 or 4 arguments:

```
usage: newt3 <x0> <y0> <z0> [<tol>]
```

where `x0`, `y0` and `z0` are the initial guesses for x, y, and z respectively, and `tol` is an optional convergence criteria which should default to `1.0d-8`. (Please resist the urge to reduce the system to a set of two non-linear equations by eliminating one of the unknowns.) Implement the test for convergence following the `newt2` example covered in class. Like `newt2`, your program should use the `LAPACK` routine, `dgesv`, to solve the linear system arising in the Newton iteration. Your program should trace the Newton iteration to standard error (again, as `newt2` does), mostly to aid you in determining when you have implemented the algorithm correctly. The only output to standard output should be the final estimate of the root (three numbers, x, y, z, on one line). Test your program by finding a root near $x = 1.0$, $y = -1.0$, $z = -0.25$ and recording what you find in `~/hw2/a5/README`. Roughly map out the "basin of attraction" of the root you locate, and report your findings in the `README` file.

Problem 6: Port your `newt3.f` code from the previous problem to the Cray J90, `charon.cc.utexas.edu`. As discussed in class you can `rlogin` directly to the account `phas761` from your account on `einstein`. Install, compile and test your code in the directory `~/<username>/hw2/a6` on the `phas761` account, where `<username>` is your username on the SGIs. Note that you are sharing this account with your classmates, so be careful not to trash files you don't own, etc. Also note that you should be careful in adhering to the coding conventions discussed in class (particularly in regard to `real*8` constants), so that you can convert to Cray Fortran using the `f77transcray` script. Leave the executable `newt3` in the `hw2/a6` sub-directory so that I can check it myself.