PHYS 210: Introduction to Computational Physics Fall 2009 Homework 1

Due: Tuesday, September 29, 5:00 PM

PLEASE report all bug reports, comments, gripes etc. to Matt: `choptuik@physics.ubc.ca`

Please make careful note of the following information and instructions, which will generally apply to subsequent homeworks as well:

1. Please do *not* be put off / terrified etc. by the length of this homework handout, including this preamble. As previous students of my more advanced computational physics courses can attest, I tend to spell things out in gory detail, so there really isn't as much work to do as a 9 page handout might suggest!
2. As we will discuss in the September 15 lab, I have created directories for all of you of the form `/phys210/$LOGNAME`, where `$LOGNAME` is the name of your `hyper` account.
3. Within `/phys210/$LOGNAME`, I have also created sub-directories `hw1, hw2, hw3, hw4` and `hw5`, which you will use to complete the five homework assignments in this course. In particular, for this assignment, you will create various directories and files that will need to reside within `/phys210/$LOGNAME/hw1`, and any reference to directory `hw1` below is implicitly a reference to `/phys210/$LOGNAME/hw1`.
4. The `hw[1-5]` sub-directories are *read, write and execute* protected from other users. *Please do NOT change the permissions on those sub-directories. This will ensure that none of your fellow students—or anyone else except myself and the TAs—can access your homework.*
5. Follow the instructions that accompany every question very carefully. Attention to detail is an important aspect of computational science, as is the ability to work precisely to specifications. Pay special attention to the name of files that you are to create, and to the ultimate locations (i.e. directories) in which they are to reside.
6. Although it is recommended, you do *not* necessarily have to use `hyper` to complete the homework: if you wish, you can use another Unix/Linux system to which you have access to do one or more of the questions. Again, however, it is your responsibility to ensure that, ultimately (i.e. by the due date/time) all files and directories that you are requested to create do indeed exist `on hyper` and in their proper locations.
7. As you complete this homework, you will need to access (and perhaps make copies of) various files/directories that reside in the account `phys210` on `hyper`. Recall that ∼`phys210` is a reference to the home directory for `phys210`, and you should have the appropriate access (permissions) for any of the needed files/directories. Let me know ASAP if you find that this is not the case.
8. Your grade *may* be adversely affected if you do not strictly follow the above instructions, in addition to those given in the individual problems below: we will willing to give you a little leeway at the beginning of the course, but will tend to be less and less forgiving as time goes on!
9. Given that many of you apparently do not have much (if any?) experience with programming (at least according to the Initial Survey you completed on the first day of class), Problems 6 and 7 are likely to be especially challenging. However, in addition to the coverage of shell programming in the lectures, we can spend some time in the labs practicing some of the basics of writing scripts, with an aim to minimize the frustration you may encounter in finishing those questions. In addition, the TAs and myself will be available for one-on-one assistance as necessary.
10. Note that the marking scheme (i.e. how much each question is worth) has purposefully *not* been included here. This homework will give the TAs and myself vital information concerning what we should expect from the class as a whole, and I don't want to unnecessarily discourage anyone at this stage. That means for example, that just because Problem 7 may be more difficult for you than Problem 6, it doesn't necessarily mean that it will be worth more marks.
11. **IMPORTANT!!** Feel free to contact me (choptuik@physics.ubc.ca) *immediately* should you have any questions about these instructions, or if you are having undue difficulty with any part of the homework. And again, you are free to seek help during the lab sessions from both the TAs and myself, as well as from myself during my official office hours (1:00-2:00 PM, Mon & Wed), or on a drop-in/appointment basis. Bear in mind though, that as much as possible, we of course want to encourage you to "think and do for yourself"!

Problem 1a:
In your `hw1` directory (i.e. `/phys210/$LOGNAME/hw1`), create a sub-directory `a1` (i.e. `/phys210/$LOGNAME/hw1/a1`). In that directory (`hw1/a1`), and using the Unix/Linux text-editor that you have chosen from the list discussed in class (i.e. `kate`, `gedit`, `gvim`, `xemacs`, `vi/vim` or `emacs`), create a file named `apple` that contains the following text from *Gravitation*, by Misner, Thorne and Wheeler. Try to duplicate the spacing, line breaks, punctuation etc. as closely as possible.

```
Once upon a time a student lay in a garden under an apple tree reflecting
on the difference between Einstein's and Newton's views about
gravity.  He was startled by the fall of an apple nearby.  As he
looked at the apple, he noticed ants beginning to run along its
surface. His curiousity aroused, he thought to investigate the
principles of navigation followed by an ant.  With his magnifying glass,
he noted one track carefully, and, taking his knife, made a cut in the apple
skin one mm above the track and another cut one mm below it.  He peeled
off the resulting little highway of skin and laid it out on the face
of his book.  The track ran as straight as a laser beam along this
highway.  No more economical path could the ant have found to cover
the ten cm from start to end of that strip of skin.  Any zigs and
zags or even any smooth bend in the path on its way along the apple
peel from starting point to end point would have increased its length.

   ``What a beautiful geodesic'', the student commented.
```

Problem 1b: Create a file in the same directory (`hw1/a1`), called `kumquat` that is identical to `apple` except that all occurrences of the word "apple" are replaced with "kumquat". Leave a brief note in a file called `README` (again in the same directory) that describes how you created `kumquat` (including which editor that you have used) and how you made the changes.

Problem 2: I have created directories for each of you that you may use to "publish" Web pages (related to this course) via my research group's web server (`http://laplace.physics.ubc.ca`). Your personal Web directory on the server (which I'll subsequently refer to simply as your Web directory) is `/phys210/$LOGNAME/public_html`, and in that directory you will find a text file `index.html` which currently contains the name of your account (i.e. the text `$LOGNAME`, and nothing else).

I have also created a "template" homepage in `/phys210/phys210/public_html/index.html`, and which you can view using a browser by going to `http://laplace.physics.ubc.ca/~phys210`.

Copy this template `.html` file into your Web directory (use the same name—`index.html`—so you will overwrite the existing `index.html`) and modify it to reflect your name, academic address (or home address if you so wish), phone-number etc. You can use a Web authoring tool (such as the `composer` component of the `seamonkey` browser), or should you want to write the HTML "by hand", use your text editor of choice. If you want to pursue the latter option, then you may find the information accessible through the *Creating Web Pages* section of the *Online Course Resources* page to be of use. (Observe that the last problem of this homework *will* require you to know a little about HTML tags, but, should you wish, you can defer learning about that until you start that question.)

If you don't feel like publishing any specific piece of information, specify it as "unlisted". Below the horizontal rule (line) in the template file, delete the text "Insert links ... please!)", and add suitably labelled links to (a) the course home page (b) the instructor's home page and (c) at least 5 Web pages that have something to do with the topic listed beside your name on the Web page `http://laplace.physics.ubc.ca/210/hw1/topics.html`. Note that the 5 links should be associated with the physics/astronomy context of the topic, since some topics may be relevant in other fields. Also, if possible, include in your web page an image (with proper attribution of its source, if you can find it) related to your specific topic.

Check your work by verifying that you can view your creation by directing your browser to our main course page, selecting *Student Pages* and then your name. Also, please send me e-mail should the way I have listed your name in the *Student Pages* list need changing.

Problem 3: Make the directory `hw1/a3`, and in that directory, create a file called `stripped` whose contents are identical to ∼`phys210/hw1/prob3/input` except that all lines that have a 'c' or 'C' in the first column have been removed. Leave a brief note in `hw1/a3/README` that describes how you solved the problem. Also, how many lines were removed from `input`, and how did you figure that out? Again, answer in `hw1/a3/README`.

Hint: The first part of this problem can be done quite easily with `grep`.

Problem 4a: Make the directory `hw1/a4`. On `hyper`, the file `/usr/share/dict/words` contains a list of "words" (mostly genuine English words—for the purposes of this question, any entry in the file will be deemed a "word"), one per line. How many words does the file contain? Answer in `hw1/a4/README`. In `hw1/a4` create files with the following names and contents (words should appear one per line):

- `23letter` which contains all the twenty-three-character (twenty-three-letter) words in `/usr/share/dict/words` in alphabetical order.
- `6letter` which contains, in alphabetical order, all the six-character (six-letter) words in `/usr/share/dict/words` that begin with 'e' or 'E', and end with 'x'.
- `r4letter` which contains all the four-letter words in `/usr/share/dict/words` in *reverse* alphabetical order (*Hint: Use the Unix/Linux* `sort` *command: as usual, type* `man sort` *to get detailed usage information).*
- `5vowels` which contains all the words in `/usr/share/dict/words` which contain all five of the vowels 'a', 'e', 'i', 'o' and 'u' (Any of the vowels may occur more than once, and can appear in the word in any order).

Leave comments in `hw1/a4/README` which describe how you solved each problem.

Hints: If `5vowels` *is problematic, study the* `grep` *section of the Unix/Linux notes carefully. It should provide the necessary inspiration. Also, note that if you use* `grep` *properly, then you should not have to do anything special/extra to have the words listed in alphabetical order, since* `/usr/share/dict/words` *is itself alphabetized.*

Problem 4b: NOTE/WARNING! The following sub-question is quite challenging, and therefore strictly **optional**. You will *not* be penalized for not answering it, but will receive extra credit (even if you do perfectly in the remainder of the assignment) if you *do* solve it! In `hw1/a4` create a file as follows:

- `3vowels` which contains an alphabetical list of all the words in `/usr/share/dict/words` which contain *precisely* 3 vowels. (Define the set of vowels to be 'a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u' and 'U', and note that individual vowels *can* be repeated.)

Leave comments in `hw1/a4/README` which describe how you solved the problem.

Problem 5: Make the directory `hw1/a5`. Use the plotting program `gnuplot` (available on `hyper`, and installable on any Linux system) to produce a plot of $\exp(-4x^2)\sin(12x)$ for $-1 \le x \le 1$. Save your plot as the Postscript file `plot.ps` in the directory `hw1/a5`. *Hint:* Use `gnuplot`'s extensive on-line help: you may find `help plot`, `help postscript` and `help output` especially useful. There is also reference and tutorial material on `gnuplot` available via the *Graphing (XY plots)* section of the *Online Course Resources* web page.

Problem 6: Make the directory `hw1/a6`. In that directory create a `bash` script, `Backup`, which has the following usage:

```
usage: Backup file [file] ...

Makes a backup copy of each regular file (as defined by the ‘test’ command).

Backup copies will have extension .O and the user will be prompted for
overwrite if the backup already exists.
```

`Backup` accepts an arbitrary number of arguments. Valid arguments are *regular* files, as defined by the `test` command, (see `man test` for full information, and ask for help if you have difficulty understanding the documentation). Also, be sure that you name your script `Backup`, with an upper case `B`, since `backup` is a Linux command, which although not currently installed on `hyper`, will generate the following error message if you type it at the command line:

```
hyper% backup
The program 'backup' is currently not installed.  To run 'backup' please
ask your administrator to install the package 'openafs-client
```

For each argument that names a regular file, `Backup` is to create a backup copy of that file, with a name given by appending `.O` to the name of the file. (Just so there's no confusion, `.O` is .O in "typewriter font", i.e. a period followed by an upper-case letter 'o'.) If a backup copy already exists, the user must be prompted to see whether the existing backup copy should be overwritten (note that this can easily be done by simply using the `-i` option to `cp`). Also, whenever a regular file is encountered, and after it has been backed up (or the user has indicated that an existing backup copy is not to be overwritten), a "long format" listing of the file and the backup copy should be output (see the `ls` section of the Unix/Linux course notes). If the argument does *not* name a regular file, the script should output an error message as follows:

```
% Backup some-directory
Backup: File some-directory does not exist or is not a regular file.
```

but should continue to process any remaining arguments (i.e. the script should *not* exit if it encounters an invalid argument).

Recall that you make `Backup` an executable with the following `chmod` command:

```
% chmod u+x Backup
```

Here, we only add execute permission for the user (you), since the script should remain hidden from everyone else's view in any case.

The following session trace, executed as `phys210@hyper` in the directory ~`phys210/hw1/prob6`, and which demonstrates various invocations of my version of `Backup`, should make the operation of the script clear. Observe that as noted below, you may find it useful to make your own local copy of ~`phys210/hw1/prob6` in your `hw1` directory, for testing your implementation of `Backup`.

```
# Demonstrates usage of ‘Backup’ in directory /home/phys210/hw1/prob6.
# Note that you may find it convenient in coding and testing your version
# of Backup to make a copy of this directory and its contents using
# an appropriate ‘cp’ command (recall the -r option for recursive copying).
% pwd
/home/phys210/hw1/prob6

# Get a full (long) listing of the directory contents.
% ls -l
total 16
drwxr-xr-x 2 phys210 public 4096 2009-09-14 11:13 dir1/
drwxr-xr-x 2 phys210 public 4096 2009-09-14 11:13 dir2/
-rw-r--r-- 1 phys210 public    6 2009-09-14 11:41 file1
-rw-r--r-- 1 phys210 public    6 2009-09-14 11:17 file2

# Invoke the script with no arguments: the usage message is output.
% Backup
usage: Backup file [file] ...

Makes a Backup copy of each regular file (as defined by the ‘test’ command).

Backup copies will have extension .O and the user will be prompted for
overwrite if the backup already exists.

# Demonstrate that file1 and file2 are regular files ...
% test -f file1 && echo "file1 is a regular file"
file1 is a regular file
% test -f file2 && echo "file2 is a regular file"
file2 is a regular file
# ... and that dir1 and dir2 are not
% test -f dir1 || echo "dir1 is not a regular file"
dir1 is not a regular file
% test -f dir2 || echo "dir2 is not a regular file"
dir2 is not a regular file

# Invoke Backup with single argument which is a regular file.
# NOTE: For each regular file, the script must generate a long listing of
# that file and its backup.
% Backup file1
-rw-r--r-- 1 phys210 public 6 2009-09-14 11:41 file1
-rw-r--r-- 1 phys210 public 6 2009-09-14 11:42 file1.O

# Modify file1 by adding a line using echo and >> (append output redirection).
% echo "another line" >> file1
# Show the new contents of file1
% cat file1
file1
another line

# Invoke with file1, but don’t overwrite existing backup.
# Note from listing how file1 and file1.O now have different sizes (the
# 5th column in the listing is the file size in bytes).
% Backup file1
cp: overwrite ‘file1.O’? n
-rw-r--r-- 1 phys210 public 19 2009-09-14 11:42 file1
-rw-r--r-- 1 phys210 public  6 2009-09-14 11:42 file1.O
```

```
# Repeat the above, but this time force the overwrite of the existing backup.
# Now file1 and file1.0 have the same sizes.
% Backup file1
cp: overwrite ‘file1.0’? y
-rw-r--r-- 1 phys210 public 19 2009-09-14 11:42 file1
-rw-r--r-- 1 phys210 public 19 2009-09-14 11:43 file1.0

# Remove file1.0 and get a full listing of the current directory.
% rm file1.0
rm: remove regular file ‘file1.0’? y
% ls -l
total 16
drwxr-xr-x 2 phys210 public 4096 2009-09-14 11:13 dir1/
drwxr-xr-x 2 phys210 public 4096 2009-09-14 11:13 dir2/
-rw-r--r-- 1 phys210 public   19 2009-09-14 11:42 file1
-rw-r--r-- 1 phys210 public    6 2009-09-14 11:17 file2

# Invoke Backup on all the non-hidden files and directories in the current
# directory using the * wildcard (globbing).
% Backup *
Backup: File dir1 does not exist or is not a regular file.
Backup: File dir2 does not exist or is not a regular file.
-rw-r--r-- 1 phys210 public 19 2009-09-14 11:42 file1
-rw-r--r-- 1 phys210 public 19 2009-09-14 11:45 file1.0
-rw-r--r-- 1 phys210 public 6 2009-09-14 11:17 file2
-rw-r--r-- 1 phys210 public 6 2009-09-14 11:45 file2.0
```

Important! Try to make the operation of your version of `Backup` *duplicate mine as much as possible, including the details and formats of the usage and error messages, and the long format listing of the regular files and their backups.*

Recall that you can use the `-x` flag in the script header for debugging purposes:

```
#! /bin/bash -x
```

Once you have coded `Backup`, you should ensure that you test it thoroughly, using, for example, the types of invocations illustrated above.

Problem 7: Make the directory ~/hw1/a7. In that directory create a bash script, `lshtml`, which has the following usage:

```
usage: lshtml outputfile
```

The purpose of `lshtml` is to generate HTML code in `outputfile` for a listing of all of the files and directories in the directory in which it is executed. For example, consider the directory ~`phys210/hw1/prob7` on `hyper` that (originally) has contents as follows:

```
hyper% cd ~phys210/hw1/prob7
hyper% ls
a  A/  bb  BB/  ccc  CCC/  dddd  DDDD/  eeeee  EEEEE/  ffffff  FFFFFF/
```

If I then execute

```
hyper% lshtml listing.html
```

the file `listing.html` will be created, with the following contents

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Directory listing of /home/phys210/hw1/prob7</title>
</head>
<body>
<h2>Directory listing of /home/phys210/hw1/prob7</h2>
<br>
<ul>
   <li><a href=a>a</a></li>
   <li>A</li>
   <li><a href=bb>bb</a></li>
   <li>BB</li>
   <li><a href=ccc>ccc</a></li>
   <li>CCC</li>
   <li><a href=dddd>dddd</a></li>
   <li>DDDD</li>
   <li><a href=eeeee>eeeee</a></li>
   <li>EEEEE</li>
   <li><a href=ffffff>ffffff</a></li>
   <li>FFFFFF</li>
   <li><a href=listing.html>listing.html</a></li>
</ul>
</body>
</html>
```

Note that the individual files and directories are listed in the form of an unordered list (see the image of the file rendered in a browser on page 9), which starts with the HTML tag `<ul>` and ends with `</ul>`. Each item in the list begins with the tag `<li>`, and ends with `</li>`. In addition, for those files which are *regular files*, once again, as defined by the `-f` option to `test`, the name of the file is surrounded by an *anchor*, which creates a link to the file itself. Such an anchor begins with a tag of the form `<a href=filename>`, where `filename` is the name of the file, and ends with `</a>`. One specific example in the above HTML code is:

```
   <a href=eeeee>eeeee</a>
```

where `eeeee` is a regular file.

When the HTML file is viewed in a browser, which can be done simply by entering into the URL type-in box the absolute pathname of the file—`/home/phys210/hw1/prob7/listing.html` in this case—the text enclosed in such an anchor will be highlighted as a link, and clicking on the link will display the contents of the file, should those files be plain-text as `a`, `bb`, `ccc`, `dddd`, `eeeee` and `ffffff` are.

Files which are *not* regular files according to `test`, are simply listed within a set of `<li>` and `</li>` tags. Again, one example from the above HTML code is

```
   <li>FFFFFF</li>
```

where `FFFFFF` is a directory.

Your script should ensure that `outputfile` does not already exist: if it does, it should print an error message as shown in the following:

```
hyper% pwd
/home/phys210/hw1/prob7

hyper% ls
a   bb   ccc   dddd   eeeee   ffffff   listing.html
A/  BB/  CCC/  DDDD/  EEEEE/  FFFFFF/

hyper% lshtml listing.html
lshtml: listing.html already exists. Remove it and reexecute if desired.
```

and then exit, without doing anything else: most importantly, it should *not* modify `outputfile` (`listing.html` in this example).

Notes/hints:

1. It may be helpful for you to know a little about the various HTML markup tags that your implementation of `lshtml` needs to generate, including
   - `<html>` and `</html>`
   - `<head>` and `</head>`
   - `<body>` and `</body>`
   - `<title>` and `</title>`
   - `<h2>` and `</h2>`
   - `<ul>` and `</ul>`
   - `<li>` and `</li>`
   - `<br>`

   Refer to the subsection *2. Doing it "by hand" ...* in the *Creating Web Pages (HTML documents)* of the *Online Course Resources* page for some links that should be of use in this regard, but note that you may be able to figure out what to do simply by studying the above example.
2. Recall that the `>>` redirection directive can be used to *append* text to a file.
3. Observe that there are several ways of getting a script to output text to standard output (which can then be redirected to a file as necessary). These include
   - Using the `echo` command, as in

     ```
     echo "Text to be output."
     echo 'Text to be output.'
     echo  Text to be output.
     ```

All three of the above forms are equivalent since there are no characters that have special meaning to `bash` in `Text to be output.` In general, however, it is safest to use quotes (usually the single quote ') around the text to be echoed.

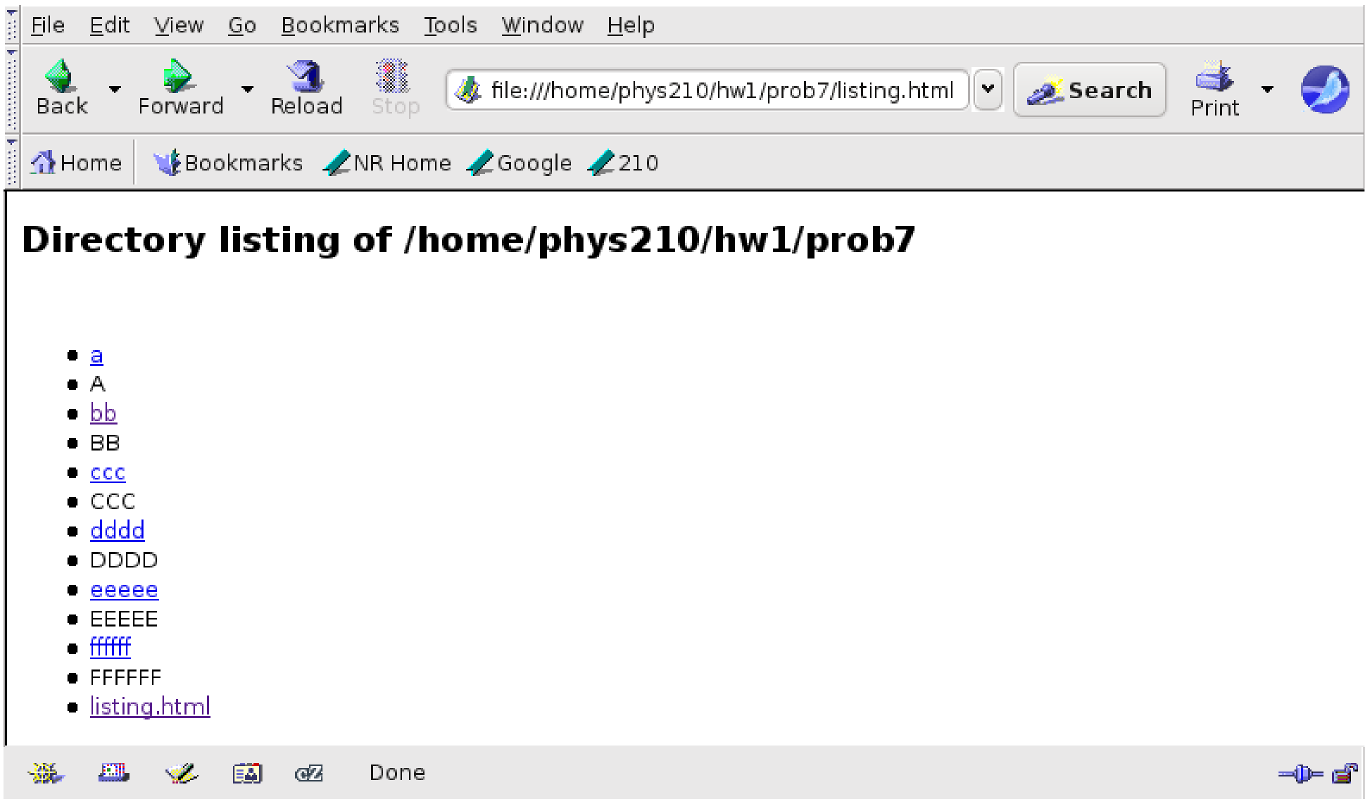
- Using the `cat` command in conjunction with a *here document*. For example

```
cat<<END
If these lines are included in a script, then when the script executes, the text
starting with 'If these ...' and ending with ' ... appear on the terminal.' will
appear on the terminal.
END
```

Also recall that shell variable constructs, such as `$var`, are evaluated in a here document.

Important! As was the case in the previous problem, try to make the execution of your script mirror my implementation as closely as possible.

Finally here's a screenshot showing `listing.html` as interpreted by the `seamonkey` browser. Note how the absolute pathname to `listing.html` appears in the URL box, how the regular files have links associated with them, and how the page heading names the directory in which `lshtml` was executed.


File Edit View Go Bookmarks Tools Window Help
Back
Forward
Reload
Stop
file:///home/phys210/hw1/prob7/listing.html
Search
Print
Home
Bookmarks
NR Home
Google
210
Done

Directory listing of /home/phys210/hw1/prob7

- a
- A
- bb
- BB
- ccc
- CCC
- dddd
- DDDD
- eeeee
- EEEEE
- ffffff
- FFFFFF
- listing.html