CSC104, Assignment 1, Summer 2006
Due: Thursday June 8th, 11:59 pm

Anastasia Bezerianos, Danny Heap

SCAVENGER HUNT AND FINGER EXERCISES

The following exercises are intended to introduce some basic skills you’ll need in this course. One of the trickier skills is finding useful instructions for carrying out a computer task. Normally you would ask a friend, or the person sitting beside you, but for this assignment this won’t be allowed. You must find your instructions from the sources mentioned in the exercise or one of the following sources:

- Asking a CSC104 TA or the instructor.
- Reading documentation included with the software (e.g. the “help” menu).
- Reading the on-line “man” pages (see below).

Some of these sources of documentation are difficult to understand, and part of your job will be to understand them (at least long enough to carry out the exercise).

1. Log in to your CDF account. Some information on how to do this will be included in lecture and the course web page. If you have access to email other than on CDF, you may write for help to admin@cdf.toronto.edu.

2. Create a directory called A1, by typing mkdir A1, then the enter key, in a terminal (for documentation, type man mkdir).

3. Change your current working directory so that it is A1 (type cd A1, then enter, and verify by typing pwd, in a terminal), and start the editor called scite. You may use the general system menu (probably at the lower left, left-click on Programs/editors/scite) or type scite (case-sensitive), then the enter key, in a terminal. Once you have scite running, open a new file (click on the file menu in scite’s application menu), type the current date in the text area, and then save the file as journalA1. For all the remaining exercises, you must record the heading for each exercise, the date and time, what you tried, what worked, what didn’t work, and any of your observations, in journalA1. For your assignment solving and journalA1 reporting try to follow the problem solving process discussed in class.

4. At a terminal type touch file1 and push the enter key. This creates a file named file1.

Now type ls (that’s a lower-case ell, not a 1), and push the enter key. Then type ls -l -t, and push the enter key. Compare the results and read about the first 65% of the output of man ls until you can explain the difference between the output of the two ls commands. Your (short) explanation should, of course, go in journalA1.
5. At a terminal type the command
   `grep 'top' /usr/share/dict/words` and then
   `grep -w 'top' /usr/share/dict/words`. The file `/usr/share/dict/words` is a huge list of English words
   that spelling programs use. Looking at the two outputs and using `man grep` try to explain the results.
   The second part of the grep command (ie 'top') is called a pattern. Patterns can be either simple
   strings (words) as in this case, or regular expressions. A regular expression is a pattern that describes
   a group of strings. Regular expressions are constructed like arithmetic expressions, by using various
   operators to combine smaller expressions.

   Type the following commands and using the output and the description of regular expressions in `man grep` (under the heading "REGULAR EXPRESSIONS"), try to explain the result. (This may be one of
   those times you should get some help from a TA or the instructor). Note the subtle difference between
   the letter ell "l" and the numeral one "1".

   (a) `grep -w '[a-l][mrt]' /usr/share/dict/words`
   (b) `grep -w 'c.l' /usr/share/dict/words`
   (c) `grep -w 'c.*l' /usr/share/dict/words`
   (d) `grep '1\{2\}l' /usr/share/dict/words`

   After finishing the above tasks, try to find the grep command that will look for all 4 letter words in
   `/usr/share/dict/words` that contain only the letters a, d, f, e, l (that's a letter ell "l", not a numeral one
   "1"). Report all your observations and findings in `journalA1`.

6. Start Firefox (a web browser) by clicking on the appropriate icon. In the top right corner there's
   a box with a "G" beside it for typing searches to Google. Experiment with typing choices of three
   legitimate English words, for example: yellow moon rise, or "yellow moon rise", to see which finds the
   fewest matches. Your chosen words should get a match of 20 or lower (but not 0). Words that appear
   as underlined in the light blue Google bar are legitimate words. Summarize the results of your search
   and observations in `journalA1`.

7. Type the command `cd` and then `enter`, to ensure that you are at the top of your home directory. Then
   create a directory called `public_html` (notice the underscore). Type `ls -l`, and pay particular attention
   to information corresponding to your new directory. Now type `chmod og+x public_html` and `enter`,
   type `chmod og+x` followed by `enter`, and and then repeat `ls -l` and `enter`. Read the output of `man chmod` until you can explain (in `journalA1`) what this command achieved (noting your observations
   in `journalA1`, of course).

8. In the directory `public_html`, create a file called `flypaper.html`, using the `scite` editor. Use `scite` to
   make sure this file contains the following:

   (a) `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">` at the beginning.
   (b) `<html>`, with a matching `</html>` at the end.
   (c) `<head>`, followed by `</head>` somewhere between the opening and closing `<html>` tags.
   (d) `<title>`, followed by `</title>`, somewhere between the opening and closing `<head>` tags.
   (e) `<body>`, followed by `</body>`, somewhere between the closing `</head>` tag and the closing
       `</html>` tag.
   (f) `<h1>`, followed by `</h1>`, somewhere between the opening and closing `<body>` tags.
Fill in the space between <title> and </title> with your full name. Between </title> and </h1> write:

```html
<p> <a href="http://www.utoronto.ca"
    img src="http://www.cs.toronto.edu/DCS/Images/blucrest.jpg"
    alt="UofT" height="18" width="20" /></a></p>
```

Inside the <h1> and </h1> tags write "Created by " and your full name. Between </h1> and </body> write:

(i) <p> and </p>. Between them add a few meaningful sentences.

(ii) another set of </p> and <p>. Between them add the three words you came up with in your google search in Ex.6.

Save your file, exit scite, make public_html your working directory, and from a terminal type chmod a+r flypaper.html. Any comments or observations you might have should again go in your journalA1.

**GCD EXPLAINED**

The greatest common divisor (GCD) of two non-negative integers (whole numbers) \( a \) and \( b \), is the largest non-negative integer that divides both \( a \) and \( b \). For example, \( \text{GCD}(3,5) = 1 \), \( \text{GCD}(12,60) = 12 \), and \( \text{GCD}(12,90) = 6 \). A special case is when one of the numbers is zero: \( \text{GCD}(115,0) = 115 \). See the lecture notes from week one for some discussion.

The greatest common divisor is useful for reducing fractions to lowest terms. Consider for instance \( \frac{42}{56} = \frac{3\cdot14}{4\cdot14} = \frac{3}{4} \), where we cancelled 14, the greatest common divisor of 42 and 56.

An efficient method for calculating the GCD of two numbers is called the "Euclidean algorithm". An algorithm is a series of steps you can follow in order to solve a problem (similar to a cooking recipe). This particular algorithm allows us to calculate the GCD of two numbers, without having to list all the divisors of both numbers.

**Euclidean Algorithm:**

Let's say we have two numbers \( a, b \) with \( a > b \). We divide \( a/b \). If the remainder \( c \) of the division \( a/b \) is 0, then \( b \) is the GCD. If not, we divide \( b \) by the value of the remainder \( c \). If the new remainder \( d \) of \( b/c \) is 0, then \( c \) is the GCD. If not, we divide \( c \) by the new remainder \( d \). If the newest remainder of \( c/d \) is 0, than the new remainder \( d \) is the GCD, etc.

In general:

\[
\begin{align*}
  a/b & \text{ gives a remainder of } c \\
  b/c & \text{ gives a remainder of } d \\
  c/d & \text{ gives a remainder of } e \\
  \ldots
\end{align*}
\]

\( w/x \) gives a remainder of \( y \)
\( x/y \) gives no remainder, so \( y \) is the GCD

**Example:**

We will find the GCD of 36 and 15.

1. Divide 36 by 15 (the greater by the smaller), getting 2 with a remainder of 6.
2. Then we divide 15 by 6 (the previous remainder) and we get 2 and a remainder of 3.
3. Then we divide 6 by 3 (the previous remainder) and we get 2 with no remainder. The last non-zero remainder (3) is our GCD.
Computing GCD in Gnumeric

Be sure to record what steps you take, what works, and what doesn’t work in journalA1.

Open the spreadsheet Gnumeric, either by typing gnumeric in a terminal, or from the general system menu. Across the first row, type the following titles for separate columns:

- Dividend (corresponding to numbers that we divide, like 36 in the first step of our example from the previous section).
- Divisor (corresponding to numbers we divide the Dividend numbers by, like 15 in the first step of our example in the previous section).
- Remainder (corresponding to the remainder of the divide of Dividend by Divisor).

To make your table more readable, with one of your headings highlighted click on Format/column, follow the arrow and select Auto-fit selection. Before continuing, save your spreadsheet as gdc.gnumeric.

In the next row, type the number 36 in the column headed by Dividend. This initial Dividend is one of the two numbers that you want to calculate the GCD of. Type 15 in the column headed by Divisor. This initial Divisor is the second number.

In the column headed Remainder you must express the remainder of the division as a Gnumeric formula. Suppose the initial Dividend is in cell A2 and the initial Divisor is in B2, then the Gnumeric formula formula:

\[=A2 + B2\]

...will give you the sum of the numbers in A2 and B2. This formula isn’t right (it does not give you the remainder as discussed in the algorithm), so you’ll have to fix it up. To guide your work, you should probably read the Gnumeric manual (click the “Help” menu in Gnumeric) section on “Working with data” and specifically look for the entry on MOD.

So far you have created the first step of our algorithm, where the two numbers are entered. Now you also need to express the sequence of the algorithm steps in the previous section as Gnumeric formulas. In the cell directly under the initial Dividend number (32) you need to type a formula that tells us the new Dividend. This new Dividend (as seen in our examples) comes from the previous step in our algorithm. If for example our new Dividend was the same as our old Dividend (it is not!), then the cell underneath 32 would have to say

\[=A2\]

Similarly, in the cell directly under the initial Divisor (15) you need to type a formula that tells us the new Divisor in the next step. Finally, you need to compute the Remainder for this step of the algorithm (Remember, you have already done this in the line above for that step).

With those formulas in place, you can copy a formula cell into the next 10 or more cells below it (click on the cell, put your cursor over the bottom-right corner, and drag). Do this with all three formulas to get an idea of how Dividends and Divisors change as the algorithm progresses.

If all went well, with initial numbers 32 and 15, you should be getting a Remainder of 0 in 4 steps or so. What is the GCD? Is it the same with our example? You can play with the parameters a bit, by making changes to the initial numbers, and see how the steps in the algorithm are affected and how the GCD changes.

Of course, “seeing” the changes of Dividend/Divisor numbers in each step is not as compelling as seeing a graphical representation. As an optional feature for this assignment, you may explore making a graph of the Dividend and Divisor numbers (see Insert/Chart).
MOLDY YOGURT

You have an old container of yogurt sitting in the fridge getting moldy. By carefully reading the manufacturer's claim printed on the side of the container, you realize that the maximum quantity of mold that could grow on this container is 1 gram. The label on this yogurt container is unusually informative. It also says that at 4°C (your fridge's temperature) you could expect a new crop of mold to replace the current crop of mold every 24 hours, and that the growth rate is \( r = 2 \).

That information puzzles you, so you phone up the yogurt producers head office to have it explained. After listening to some helpful muzak and moving through the phone tree, you talk to a customer service representative who explains that the growth rate tells you how big the next crop of mold will be compared to the current one. If the current crop of mold weighs \( w \) grams (and, according to the manufacturer, \( w \) is somewhere between 0 and 1), then the old crop of mold generates new mold at the rate of \( wr(1 - w) \).

When you ask about the factor of \( (1 - w) \), the service representative explains that as you get closer to the maximum amount of mold the container can support, the rate of reproduction is reduced.

After doodling on the back of an ATM slip, you realize that if my current crop of mold weighed \( w \) grams, in 24 hours you should expect \( wr(1 - w) \) grams. Then you could repeat the calculation with the new population to figure out how much mold I'd have 48 hours later, 72 hours later, and so on.

MOLD ON A SPREADSHEET

In order to get a handle on the moldy yogurt situation, you should probably model it with a spreadsheet.

Use gnurneric to create a spreadsheet called mold.gnurneric. Create the following headings for columns:

- Weight
- Rate

In the cell underneath Weight, put an initial weight of 0.0000, where 0000 is the last four digits of your student number. This is the initial weight of mold in your yogurt container. In the cell underneath Rate, put 2. This is the reproduction rate, corresponding to the symbol \( r \) in the last section.

Rather than carrying out calculations with a pencil on the back of an ATM slip, let gnurneric calculate how much mold you'll have each day for a hundred days or so. You'll need to write a formula that calculates how much mold there will be in 24 hours in the cell underneath the initial weight of mold. For example, if the amount of mold always doubles in 24 hours and your initial weight is in cell A2, then underneath it would write \( = 2 \times A2 \). Of course, that's not the right formula, since you know that the formula from the previous section involved the rate, \( r \). Figure out the right formula, and then copy it down the column (in the same way as you did for the GCD) for a hundred days or so. Hint: If your reproduction rate is in cell B2, then the formula will use \( \$B2 \). You'll need to understand (and explain) the use of dollar signs ("\$") in the formula.

If you've got the formula right, after a few days the mold population settles at about 0.5 grams. Try setting the rate higher, say to 2.5, 2, 3, 3.5, or 4. What sort of patterns do you see? A rate of less than 1 is problematic (what happens?). A rate of more than 4 is also problematic, and the explanation for this goes beyond the scope of this course. You may want to create a graph of the mold populations over time.

WHAT TO HAND IN

Under Assignments on the course web page you will find a link to the CDF submit facility. Submit the following files:
• journalA1
• gcd.gnumeric
• mold.gnumeric
• flypaper.html

You should submit your files early and often. The first time you create a file with meaningful content, submit it. You may re-submit the same file as many times as you wish, and only the last submission is stored. A good habit is to re-submit your files each time you improve them.