CSC104, Assignment 1, Summer 2006
Due: Thursday June 8th, 11:59 pm
Anastasia Bezerianos, Danny Heap

SCAVENGER HUNT AND FINGER EXERCISES

Here I masquerade as a new user.

1. Log in to your CDC 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.

   SOLUTION: Here’s what I see when I log in from my machine. Your mileage may vary.

   werewolf:~% 

   When I log in on one of the CDF terminals, I see a graphical environment with various icons I can click on, and a terminal where I can type commands.

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

   SOLUTION: Here’s what I see when I type ls -l from my main (home) directory on CDF:

   werewolf:~% ls -l
   total 4
   drwx------ 2 t6bezeri ta 512 May 18 17:31 A1
   drwx------ 3 t6bezeri ta 512 Jan 10 18:10 Desktop
   drwx------ 5 t6bezeri ta 512 May 18 17:31 Winter2006
   drwx------ 2 t6bezeri ta 512 Jan 19 11:11 mail
   werewolf:~% 

   Notice the A1 at the top, and at the extreme left of that line the “d”, which indicates a directory.

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.
**Solution:** Here's my first journal entry:

June 1, 11:59 am
I honestly tried to start the assignment on time, but once again I am soooo late.

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.

**Solution:** Okay, I touch this file1 whatever that may mean. I don't get any errors, so apparently the system knows more than I do about this.

Here's what I get for those ls commands. The leading "-" is an alias for my home directory.

```
werewolf:~/A1% ls
file1  journalA1
werewolf:~/A1% ls -l -t
```

```
total 1
-rw------- 1 t6bezeri ta 0 May 30 13:06 file1
-rw------- 1 t6bezeri ta 20 May 30 13:04 journalA1
werewolf:~/A1%
```

The first version lists the file name journalA1 and file1. (Mystery solved! Touch actually created the file file1.)

The second version gives a lot of other information, so I type "man ls". The man page says that the "-l" gives a long listing, but no details of what "long" consists of. I can see my username and a date and time, although I am not sure about the other stuff. The man page also says that "-t" prints the files based on the last time they were modified. After a bit of squinting I see that the time indication for file1 is actually more recent than that of journalA1. To verify this I type "ls -l" (yes, no -t) and get exactly the same thing. What is going on? Since I know how to create new files I do "touch somefile", "touch anotherfile". I try my commands again and lo and behold, without the -t option is presents the files in alphabetical order. It was just bad luck that both the alphabetical and time order of my initial files was the same.

There is still something bothering me about all this. In the ls -l -t output I get a "total 1" indication. I would expect that this should be "total 2" since I have two files, and 4 after touching files like crazy. I ask the TA about this (we are in good terms) and she says my hunch was correct, it is just that all the files I have created with touch are actually empty (they have 0 size) and are not counted. I see a 0 in the long listing of file1, so that number before the date is probably the size of the file (another small victory for me).

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][a][m][r][t]' /usr/share/dict/words
(b) grep -w 'c.1' /usr/share/dict/words
(c) grep -w 'c.[^1]' /usr/share/dict/words
(d) grep '1\{2\}' /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 journalAI.

SOLUTION: I type the two grep commands and this is what I get.

werewolf:~/A1% grep 'top' /usr/share/dict/words
Aristophanes
Aristophanes's
....
utopian
utopians
utopias

werewolf:~/A1% grep -w 'top' /usr/share/dict/words
top
top's

Well, one thing is clear, I get a list of words. In the first case the list is very long and it seems that all the words contain "top". In the second case only the word top is in the list. I check the man page for grep and confirm my suspicions. Grep shows all the lines that contain a pattern (in my case the word "top"). It seems that in the "words" file each line is a word. The "-w" option seems to restrict grep to whole words (thus the smaller list in the second case with variations of top).

I read the description of Regular Expressions and I believe I understand that they are a way to represent multiple words and "top" is just a very simple (and not very general) regular expression. The man page goes into stuff about "extended" and "perl" and I get lost. I'll try to solve the rest of the question and refer back to the man page for specific stuff (no way I am reading the entire thing!).

(a) I type grep -w '[a-l][a][m][r][t]' /usr/share/dict/words and this is the result I get:

werewolf:~/A1% grep -w '[a-l][a][m][r][t]' /usr/share/dict/words
bar
bar's
bat
....
lam

3
Well, one thing is for sure, this grep call returned whole 3 letter words. It also seems that all of them have "a" as their second letter. That would explain the second "a" in '[a-l][mrt]'.

Okay, now to figure out the "[l]". The man page says that a bracket expression (something inside "[]") matches any single character in that list. So the three letter word should end in either "m", "r" or "t". Yep, I checked all of them and they do. The "[a-l]" part seems more complicated. Apparently this is called a "range expression" and basically represents all letters between "a" and "l".

So all these three letter words start with a letter of the alphabet between "a" and "l", have "a" as the second letter, and end in "m", "r" or "t". Although "jar" is a word that is described by the regular expression '[a-l][mrt]', the word "top" is not, because it breaks all the rules (starts with a letter bigger than "l", the second letter is not "a", and does not end in "m", "r" or "t").

(b) I happily type grep -w 'c.l' /usr/share/dict/words. Just by glancing at it I am guessing it will select words starting with "c" and ending with "l". I get:

```
werewolf:~/A1% grep -w 'c..l' /usr/share/dict/words
call
cell
cell's
cull
curl
```

```
werewolf:~/A1%
```

Okay, I was correct and all words start with "c" and end in "l". It seems though that they are all 4 letter words (can't seem to find anything else in common). I check the man page on regular expressions and find out that "." matches any single character. So the 2 "."'s plus the "c" at the beginning and "l" at the end give all the four letter words starting with "c" and ending with "l".

I am getting the hang of this! I try for fun 'c.s' and get all 4 letter words starting with "c" and ending in "s", which turns out is mainly the plurals of three letter words starting with "c" .)

(c) I feel pretty good about myself with the grep situation. I then type grep -w 'c..l' /usr/share/dict/words and I get the same results as before. I am a bit tired and I take some time to realize that one of the "."'s is now an "*". I replace it and try again with grep -w 'c.*l' /usr/share/dict/words.

```
skywolf:~% grep -w 'c.*l' /usr/share/dict/words
cabal
cabal's
call
cymbal's
cynical
```

```
skywolf:~%
```

Wow! This is definitely NOT a list of four letter words. I get words like "call", "categorical", etc. Based on what I saw before in part 5b, I spot almost immediately that all the words actually start with a "c" and end with an "l", but anything in between seems fair play. The
"\"" is any possible character. I check the man page for the regular expressions, and I see that "*\" means that the character before it ("\"", or in other words any character) will be matched zero or more times. So basically I just asked for words starting with "e", ending in "l" and have anything in between.

(d) Just when I was thinking I got the hang of the grep situation, this beast comes along ... I am really careful this time when I copy the grep command `grep 'l\{2\}' /usr/share/dict/words` to my terminal.

```
$ grep 'l\{2\}' /usr/share/dict/words
```

Achilles

```
...
zillions
```

skywolf: `% grep 'l\{2\}' /usr/share/dict/words`

This "try to explain" stuff is all well and good, but I am eager to try out things with grep myself. Okay, so I have to get all 4 letter words that are made up of letters a,d,f,e,l. From part 5 I know that to match any of these characters I need to put them in brackets, so [a,d,f,e,l]. From what I saw in part 5b I need to put those brackets one after the other four times to get a 4 letter word. So I type `grep '[a,d,f,e,l][a,d,f,e,l][a,d,f,e,l][a,d,f,e,l]' /usr/share/dict/words` and get nothing like what I expected. In fact I get words of all sizes. Ooops, I forgot the "-w" parameter. I add it and try again. I am so proud of myself for getting this right!

As I am getting ready to call it a day, I remember that strange 'l\{2\}' . Is it possible it can work here too? I try `grep '[a,d,f,e,l]\{4\}' /usr/share/dict/words` and YES it works (and looks better too!)

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.

SOLUTION: The first thing I do is try the search words the exercise mentions. I get 7,580,000 hits for yellow moon rise and 17 for "yellow moon rise". These quotes are really doing something strange. I try both searches a few more times and FINALLY I spot a difference! In all the hits from the second search the words yellow moon rise are directly adjacent and in the same order. I try moon yellow rise without the quotes (6,940,000) and with the quotes (no hits!). I click away at Google until the page http://www.google.ca/help/basics.html said that the quotes force Google
to search for the exact phrase. So I guess no one ever thought of something intelligent to say about "moon yellow rise".

Okay, so I need to find less than 10 hits in my search. I will try to use quotes to help me get few results. "cat dog car" obviously is too popular and I get 196 hits, and "assignment unresolved happiness" makes no sense so I get no hits. In a flash of inspiration I try "golden moon rise" and get 11(!) search hits.

This should have been the end of that, but I want to find fewer than 20 matches without the quotes. I try to think of three unrelated words. I go as far as to search for aardvark barometric futility, but I still get about 200 hits. It seems there are many people out there who have websites with lists of words!!! I finally give up. My "golden moon rise" will have to do.

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).

SOLUTION: After I realized that I needed to go back to exercise 2 to figure out how to create a directory things got better. Here’s what I got from the two versions of the ls command:

werewolf:~% mkdir public_html
werewolf:~% ls -l
total 5
drwx------ 2 t6bezeri ta 512 May 30 13:06 A1
drwx------ 3 t6bezeri ta 512 Jan 10 18:10 Desktop
drwx------ 5 t6bezeri ta 512 May 18 17:31 Winter2006
drwx------ 2 t6bezeri ta 512 Jan 19 11:11 mail
drwx------ 2 t6bezeri ta 512 May 30 18:09 public_html
werewolf:~% chmod og+x public_html/
werewolf:~% chmod og+x ~
werewolf:~% ls -l
total 5
drwx------ 2 t6bezeri ta 512 May 30 13:06 A1
drwx------ 3 t6bezeri ta 512 Jan 10 18:10 Desktop
drwx------ 5 t6bezeri ta 512 May 18 17:31 Winter2006
drwx------ 2 t6bezeri ta 512 Jan 19 11:11 mail
drwx--x--x 2 t6bezeri ta 512 May 30 18:55 public_html
werewolf:~% After suffering a bit of eyestrain, I realized that the two outputs are not identical. The row for public_html has a couple of extra 'x' characters in the first column after I do the chmod command. The man page for chmod says that it changes the file access permissions (whatever that means). If I read a bit further, it says that some combination of the characters 'ugo' determines which users' access to the file will be changed. So I changed 'og', whoever that is. Wait, it then says that 'g' is other users in the files group, and 'o' is other users not in the files group. So I guess I've changed the access for other users (whoever they are). Apparently there's a user who owns the file (if there's any justice, this should be me), indicated by 'u', but I didn't change this. The man page says that the '+' adds some access, and the 'x' is execute (sounds grim) or access for
directories. Putting this all together, I've allowed "others" to execute or access public.html. I hope they wear hoods.

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 <body> and </h1> write:

    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 journalAI.

SOLUTION: Here's what I put in my flypaper.html:

```<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head><title>Anastasia Bezerianos</title></head>

<body>

<p>
    alt="UofT" height="18" width="20" /></a>
</p>

<h1> Created by Anastasia Bezerianos</h1>
<p>This is a meaningfull, albeit useless sentence. I could think of nothing original to write. 
    I am wondering though why I had to name this file "flypaper.html" ...</p>
<p>golden moon rise</p>
</body>
</html>
```
After getting reasonably confused with all the words in "< >" (someone said they are called "tags") I looked at my creation. There is something there called "blue.jpg" and I think that .jpg means this is the name of an image. The "height" and "width" words seem to be related to this image, although what the "alt" is I have no clue. Now I am really curious as to what it is that I have created. Than html word looks familiar too. I think it has something to do with the internet. All the other stuff about being strict with something I do not understand.

Anyway, on with the exercise. I knew that chmod was going to change some permissions, so I first did a listing, so I could see what difference it made. Here's the before and after:

```
werewolf:/public_html/ ls -l
 total 1
-rw------- 1 t6bezeri ta 3 May 30 18:55 flypaper.html
werewolf:/public_html/ chmod a+r flypaper.html
werewolf:/public_html/ ls -l
 total 1
-rw-r--r-- 1 t6bezeri ta 3 May 30 18:55 flypaper.html
werewolf:/public_html/
```

The entry for the file flypaper.html has picked up some 'r' characters on the left-hand side. The man page for chmod says that I gave read access to "all". If I'd know everyone could read it, I would have taken care to write something intelligent.

**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, GCD(3,5)=1, GCD(12,60)=12, and GCD(12,90)=6.

A special case is when one of the numbers is zero: GCD(115,0) is 115. See the lecture notes from week one for some discussion.

The greatest common divisor is useful for reducing fractions to be in lowest terms. Consider for instance \(\frac{42}{56} = \frac{3\times14}{4\times14} = \frac{3}{4}\), 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 \(b\) with 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 \(c\) with the new remainder \(d\). If the newest remainder of \(c/d\) is 0, then the new remainder \(d\) is the GCD, etc.

In general:
- \(a/b\) gives a remainder of \(c\)
- \(b/c\) gives a remainder of \(d\)
- \(c/d\) gives a remainder of \(e\)
- ...
- \(w/z\) 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 journalAI.
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 gcd.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 “Function Reference” and then ”Mathematics” and specifically look for the entry on MOD.

So far you have created the first step of our algorithm, where the two numbers are inputted. 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 (36) 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 36 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 36 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 get 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).

**Solution** I put a screenshot of the formulas I ended up with at the end of this report. Now I'll tell you how I got to that point.

In spite of a sinking feeling that this part of the assignment will turn me into an accountant, I started up Gnomic and filled in the column headings, as asked. Formatting the headings to be more readable and saving my (minimal) version of gcdgmnumeric wasn't too bad, using the pull-down menus.

Filling in the next row with 36 under Dividend and 15 under Divisor was fine. The Remainder part gave me some grief though. I first put A2 + B2 in the cell under remainder and, well the cell had "A2+B2" inside it. From all the formula discussion I would expect something more interesting. AHA, I am missing a "=". I add that to the formula and what I get is the content of cell A2 (36) added to the content of B2 (15). So the assignment said this is not the correct formula and I need to look at a MOD thingy. The help page says that MOD gives the remainder of a division between two integers. So in my cell C2 (under the Remainder heading) I type =MOD(A2,B2) and I get something reasonable for my efforts (6).

On to the next row! After spending an hour or so looking at the example I see a pattern. We always divide two numbers and get the remainder. These two numbers come from the previous step of the algorithm. The new Dividend is the old Divisor. So in my cell A3 I write =B2. My new Divider is my old Remainder. So in my cell B3 I write =C2. The Remainder should come again from dividing the two cells before it. So in C3 I write =MOD(A3,B3).

I copied each formula to the next ten cells, and they both generated errors of some kind after a point. I check the formulas for the Dividend and Divisor columns and I observe that the row numbers increase by 1 (A2 becomes A3). Great, I don't need to write those formulas again for each step of the algorithm. The errors I see seem to start after I get a solution, after I get a Remainder of 0, so I ignore them.

I create the graph of the two first columns as the assignment says. It seems that the two numbers drop off suddenly on the first step of the algorithm and then the drop becomes more gradual. The Divisor is always smaller than the Dividend, since it is the remainder of a division with the previous Dividend (which is now the Divisor).

To make sure my spreadsheet actually calculates the GCD I check that the rows created match the example given in the assignment. I also try a few numbers of my own. It seems that for different numbers the solution is found in different number of steps. For example for 150 130 I find the solution (GCD 10) in 3 steps and my graph lines are less steep at the beginning, but drop off suddenly after the third step. For 151 131 I get the solution in 7 steps (GCD 1) and my line is as steep, but its tail (the last part) is longer. It seems that the number of steps is not similar for similar numbers, since 151 and 150 are similar and 131 and 130 are too. I wonder if 151 and 131 are special in some way ...

Also I notice that the number of steps does not depend on how big the numbers are. For example I get the GCD of 100 and 900 which is (100) in 1 step!
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 \( r = 2.5 \).

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

- Weight
- Rate

In the cell underneath Weight, put an initial weight of 0.1, where X 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 gnumeric 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 \( =2*A2 \). 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 \( \$B\$2 \). 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.

Solution: Again a screenshot of the formula is at the end of this report.

This assignment is getting stranger by the minute. I must be crazy to calculate stuff about mold (I don’t even like yogurt!). I will approach this as a concerned lazy person who never cleans the fridge. You never know when the next time you need to come head to head with mold will be.
I quickly place the titles for Weight and Rate and the initial weight for the mold in A2 (0.3124 in my case) and the initial rate in B2 (2). Now to find the formula for the amount of mold the second day: I believe the representative already gave me the answer \( wr(1 - w) \), where \( w \) is the mold I have in A2 and \( r \) is the rate I have in B2. So in A3 I write \( = A2 * B2 * (1 - A2) \). There is this hint about \( B2 \).

I change my formula to \( = A2 * B2 * (1 - A2) \).

I copy both formulas 100 times and check some of the formulas in the mold column. I observe that the row numbers increase by 1 (A2 becomes A3), but that B2 stays the same down the entire Weight column! That makes some sense: the mold rate isn’t going to change over the days, but the mold weight does, so we need to change A2 to A3, etc., but we want to keep going back to B2 hence the dollar signs. So I delete all the copies in the Rate column and only keep B2.

I make a graph of my mold weight (I am getting attached to it by now). It seems that the weight gets bigger pretty fast (5 days in my case), until it reaches 0.5 and then stops. This make sense, since I can have maximum weight of 1, and if the rate of growth is 2, in another day the mold will be twice as much (1) so it stops growing. I wonder if my student number was smaller what would happen. I try an initial weight of 0.0013 and see that it reaches 0.5 in 14 days. I guess it is harder to reach 0.5 when you start from a smaller number.

I start changing the rate as the assignment asks. This is so cool! With rate 2 the graph is fairly smooth, but with 2.5 I see a spike. With 3 I see many spikes and with 3.5 the spikes get bigger! With 4 they are so big that they reach close to 0. It seems that for rates larger than 2 the weights don’t just go up each day, they go down too! With a rate larger than 4 I get negative weights for my mold (which does not make sense), and eventually I get an error.

With all the rates that are below 1 my mould instead of getting bigger gets smaller and smaller (self cleaning mould, now that’s a thought...)

**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.