CS214 Take-home exam

Due: In CMS in 24-hours. `http://cms.csuglab.cornell.edu` .
Make sure you use the proper assignment (Wed-Thu or Thu-Fri).
Problem 1: ECHELON. 50 points. Turn in `echelon.sh` , `echelon.pl` , `echelon.py` and `echelon.txt` .
Problem 2: The Origins of Life. 50 points. Turn in `download.sh` , `search` and `extract` .

Introduction

This is the take-home exam for CS214. You have 24-hours to solve it, and submit to CMS. By handing it in you are automatically agreeing to the following terms.

Since it is an examination, you are not allowed to collaborate with another person, ask or talk to anyone about its contents, or ask for assistance or help for any of the problems, including online forums or chat.

You *are* allowed to use the lecture notes, earlier homeworks, any UNIX programs. You are also allowed to search the Internet in case you need to read up on how something is done, as long as you do not directly copy code that you did not write, and that you include a reference to the site that you were looking at. Remember that I can also find code on the web.

The bottom-line is this: Use common sense. This is an exam and you're expected to follow the rules about academic integrity at Cornell (see homepage).

If you do get stuck, however, you can ask *me* via e-mail. As long as you provide enough detail about where you are in the problem and what you're stuck on, I will make an offer for a hint. The offer is the number of points that I'll take off for the problem in exchange for the hint. You can you choose to accept it or decline it as you wish. Bear in mind that I will also have other things to do during the 24 hours so don't expect an instant response.

Clarifications are different from hints, so please let me know if something is ambiguous, and I will make sure it gets clarified on the webpage and in CMS.

You should make sure that none of your scripts leave any temporary files after they finish executing.

And finally, GOOD LUCK! :-)

Problem 1: ECHELON

For a long time, conspiracy theorists have claimed that the U.S. and U.K. run *ECHELON* **[1]** system has automatically been searching for trigger words and phrases in phone conversations and e-mail communication across the globe for over two decades. Following controversial claims of industrial espionage, the European Union investigated and reported in 2001 that the existence of the system "*is no longer in doubt*", although its technical capabilities for surveillance were probably more limited than asserted by media sources.

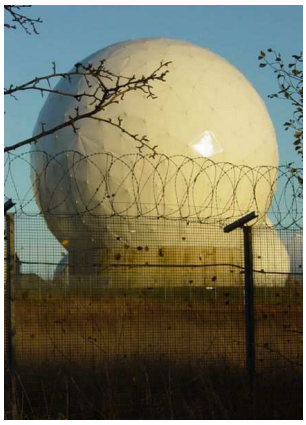

Figure 1: Suspected ECHELON station

In this exercise you will implement the search mechanism in your mini-ECHELON system. It is well known that the most precious resource on the Cornell campus is *Free Food* (FF), so the mission of the mini-ECHELON system is to gather intelligence about potential sources of FF.

We are interested in all messages that contain any of the following words, ignoring case.

- free food
- refreshment or refreshments
- catering or catered

You also don't want to approve of any messages that contain the word "pizza" due to your recent bad personal experience with that resource. A message is stored in a file that's given as the first argument to your script. If the message contains any of the words above, and does *not* include "pizza" (ignoring case), you should print "INTERCEPT", otherwise "LET GO".

Suppose *file1* contains "*Free food! Pizza at noon*", *file2* contains "*There will be free refreshments after the talk*", and *file3* contains "*Featuring CATERED FOOD from The Heights*". Your script should work as follows.

```
> ./script file1
LET GO
> ./script file2
INTERCEPT
> ./script file3
INTERCEPT
```

However, it is not clear how extendible your design must to be so you're not sure what scripting language you should be using. So you try out three of them.

1. **(15 pts)** Write a bash script *echelon.sh* that does the job.
2. **(15 pts)** Write a Perl script *echelon.pl* that does the job.
3. **(15 pts)** Write a Python script *echelon.py* that does the job.
4. **(5 pts)** Write a short paragraph describing which of the three versions you thought was most appropriate for the job, and why. What other features would you like to incorporate into your mini-ECHELON in the future? Submit this as *echelon.txt*.

Problem 2: The Origins of Life

Scientists are divided in their opinion on how organic molecules managed to organize into large protocells **[2]**. They are faced with a kind of a chicken-and-egg problem - you need proteins to replicate DNA, and you need DNA to produce proteins. The question is what bridged these two complex biochemical structures. Some believe that RNA preceded DNA as a self-replicating molecule ("*RNA world*") since it's simpler than DNA, yet it can direct chemical reactions. Others argue that smaller molecules came first and gradually became more complex via closed reaction cycles, or the "*metabolism-first*" hypothesis.

Figure 2: DNA helix

Two researchers, James Ferry and Christopher House, are investigating the energy processes of a primitive underwater microorganism called *Methanosarcina acetivorans* **[3]**. It appears that the organism harnesses energy by means of a simple reaction between acetate and iron sulfide using only two proteins, instead of the complicated protein interactions that take place to produce energy in protocells.

Enthusiastic that the results might give evidence to the metabolism-first hypothesis, they ask you to write a couple of programs to help them find *primers* for the recently sequenced genome of the microbe. This might help them identify potential locations of mutations and exploring the resulting behavioral differences.

1. **(10 pts)** Write a bash script *download.sh* that automatically downloads the *Methanosarcina acetivorans* genome, i.e. the *methanosarcina.fasta.gz* file from the site

   http://www.broad.mit.edu/cgi-bin/annotation/methanosarcina/download-sequence.cgi

   into the current directory and decompresses it using *gunzip*.

   Your script should print "SUCCESS" if the file could be downloaded and unzipped, and "FAILURE" otherwise. Nothing else should be output to the terminal.

2. **(20 pts)** Write a program called *search* using either bash, Perl or Python to search the genome you downloaded. A genome is a long sequence of nucleotides or base pairs, each of which is either A, C, T or G. This particular sequence has 5,751,495 base pairs and is stored in the FASTA format as described in **[4]**. However, really all you need to know is that lines that start with $>$ are comments, and that each line contains at most 50 characters plus a newline (`\n`).

   Your program should do the following. Given an argument, say `CATTAGATCATT` , it should search the genome in the file `methanosarcina.fasta` and tell you how many times that DNA sequence appears. The argument could be a regular expression. For the example,

   ```
   $ ./search CATTAGATCATT
   1
   $ ./search CATTAGATC.TT
   2
   ```

Also, don't worry about pairs of overlapping results overlap (like searching for `GAGA` in `GAGAGA`), those won't be tested.

Hint: Think about the problem for a little bit longer if it appears to be absolutely trivial.

3. **(20 pts)** Write a program called *extract* using either bash, Perl or Python that does the following. Given two arguments n and m with $n < m$, it prints the subsequents of the genome in `methanosarcina.fasta` that correspond to base pairs n through m, both included. The first base pair has index 0, and the last one should be 5,751,494. Don't confuse this with line numbers.

   For example,

```
> ./extract 0 0
A
> ./extract 0 10
AGTATGAAACC
> ./extract 7144 7156
TTTTTTCTTATTG
```

References

[1] Wikipedia. March 2008. *ECHELON*. http://en.wikipedia.org/wiki/ECHELON. Retrieved on 03/10/08.

[2] Michael Schirber. June 2006. *How Life Began: New Research Suggests Simple Approach.* http://www.livescience.com/animals/060609_life_origin.html. Retrieved on 03/10/08.

[3] Wikipedia. November 2007. *Methanosarcina acetivorans*. http://en.wikipedia.org/wiki/Methanosarcina_acetivorans. Retrieved on 03/10/08.

[4] Wikipedia. February 2008. *FASTA format*. http://en.wikipedia.org/wiki/FASTA_format. Retrieved on 03/10/08.