UOIT/DC COMPUTER SCIENCE CLUB ANNUAL CONTEST 2013

The following problems are an amalgamation of various challenges from different problem domains, the point being that you need to think in order to solve them. Good Luck!

Submission Guidelines

- The contest submission deadline is **January 27, 2014**
- Put your solution to each problem in a separate folder **numbered from 1 to 5**
- If your submission requires any type of setup or other work in order to evaluate it **PLEASE INCLUDE A README FILE** for each submission.
- Compress **all of your solutions** into one **zip or tar** file and name the file with your name (ie. jon_doe.zip)
- Email your solutions to **admin@cs-club.ca** with the message title **CONTEST**
- For contest hints or questions about the contest, please email us or join our mailing list: **https://groups.google.com/group/uoit_csc?hl=en**
- For information about the Computer Science Club visit us at: **https://cs-club.ca**

Contest Problems

1. Word Frequency
    a. Write a program that can create a list of the frequency of words using any **ASCII** text file as input. Your program **must produce an output file** containing the number of times each word was found in the input followed by the word. The words should be listed from most frequent **first** to least frequent **last**. Your program must treat all words as **case-insensitive** (hello and HELLO are counted as 2 instances of hello) and **only count words with alphabetic characters,** ignore any characters that are not a-z. The following is a recommended text file to test with: **http://www.gutenberg.org/files/2852/2852.txt**

    b. (**BONUS**) Most implementations, awarded to whomever provides the most implementations to the above problem. An implementation in assembly language will receive a bonus of **+10** the number of implementations (and will have a strong bias to win this award).

    c. (**BONUS**) The shortest solution based on the number of characters used to

implement the program, **new lines and whitespace are not counted as characters** (I want to still be able to somehow read it)! **Your program must be fully functional and meet all of the requirements of part a**, it should produce the exact same output as you had for part a of the problem.

2. The Case of the UOIT Email Spammer
    a. A year after the huge spam email crisis that rocked UOIT to it's very core progress has been made on identifying the culprits! After long hours of tracking several ethernet cables, Billy, a student in criminology at UOIT, made a break in the case. A DNA sample was discovered prompting a DNA test of all persons of interest related to the case. Billy has come to you with a sample of the DNA sequences of every student who was swabbed. However he made a mistake and mixed up the sample from the crime scene with the samples taken from the students. He has no way of identifying which sample is from the crime scene and which is from the group students. And so he has turned to you, an expert programmer, to prove your worth and solve the crime of the semester! He assured you that if you can **identify the two closest matching DNA sequences that they will have found the criminal.** It would also be a good idea to **list the closest matches for every other DNA sequence** as that would give them better confidence in your results. There is one caveat in all of this: due to UOIT's generous spending in other areas, a reliable DNA sequencer could not be purchased and the one used is prone to errors (DNA sequencing errors are displayed as **?** in the sequence files) so your program must be **robust to handle processing errors** when reading the DNA sequence files containing the four base pairs **(A, T, C, G)**. The following download contains the DNA sequences for each student, including the DNA sequence from the crime scene that was mixed up with the samples, find the criminal!

       DNA Sequences: **http://goo.gl/Uhk1r1**

    b. (**BONUS**) Since the DNA tests were taken from all the persons of interest, the culprit must already know progress has been made to catch them. Time is of the essence, DNA sequencing and analysis is a big challenge and finding the right solution can make all the difference! A bonus mark will be awarded for whomever comes up with the most optimal solution. By optimal I am referring to the performance/memory/algorithmic complexity, so using a faster programming language (like C instead of Python) will not make a difference.

3. Jon's Conjecture
    a. **Every** prime number > 5 can be expressed as the sum of three prime numbers. Implement a solution that given the number of prime numbers desired as input will show **the number of prime numbers > 5 specified as well as the three prime numbers that each is made up of.** For example if **given 3 as input** it would

show the following three prime numbers as output:

```
7 {3, 2, 2}
11 {5, 3, 3)
13 {5, 5, 3}
```

b. Given your knowledge acquired (**if possible**) implement a solution that could potentially be used to generate an infinite sequence of primes given the previous prime numbers (e.g. generate 7 using 3+2+2; 11 using 5+3+3, ad infinitum). Do you think this conjecture is true and will hold for **ALL prime numbers**? Try and justify your reasoning and think about it!

4. Russian Blocks
   a. The CS Club is pleased to announce the release of their new highly-polished indie video game “Russian Blocks”, complete with a high score system so you can brag to all your friends. However, the high score system was developed in a hurry, so there may be some security issues. To complete this problem you **must hold a top score on the leaderboard,** which can be found here: **http://cs-club.ca/scoreboard**. You **MUST** use your full name and CS Club account, if you forgot your password try resetting it: **https://cs-club.ca/reset** if you don’t have a CS Club account email us **ASAP**. Now download the game and get practicing!

      Russian Blocks: **http://goo.gl/upnEsn**

   b. (**BONUS**) If all the other scores on the scoreboard are zero and you are the only one with the top score you will get a bonus point.

5. Super Car Racer
   a. Using **http://boxcar2d.com/** you are able create an optimal car based on the track using Artifical Intelligence methods. For this problem you be create an optimal car using whatever parameters you wish (eg. any mutation rate, algorithm, etc.). When submitting you will take click the ‘Copy Best’ button and paste the text into your solution for problem 6.

      An example of a copied car is given:
      eNqzv7P/qUxbWpoD+79VnwQYGOzvzaiOFDY2tj+xZYbrs7Q0+9faEYt8zpx1YFR
      gEwTJ3685EvDmzBkHxuotB6WB6i58qRHrOXPW/suRlIZlaen2D767CcwB8i9cU
      HosOXOW/dPCMzsqgfSn9OVs04Dyz/jf6xgD9R3buijR2NjkPxA4iHG6Lzddz2D/0
      s+qa1laGgMDA5ODkIWs/gOpt/YPzKIKJGfOBKmzf7hKv2LC9l32X0O9p0bOnAX
      WK7r2TF9+vJj91/23y4uNjcHqvj+Z7Ve1o8L+7o9724FuAJvH7+JneC5vrv3jbPHWZ
      2A72Bx43D9/9o1eZH+Ut5AH6hb7/HQWGaPvs+xv3A0SnQnUm7tiBRjz9tzDimHy
      81a9hrNh2H6eP9AeZgCSO6LV

The track that you are going to be competing on is called **‘The Hills’**. Your objective is to obtains the highest score. Once submitted each car will be run to determine their score and the **person with the highest score wins.**