OLLSCOIL NA hÉIREANN, CORCAIGH
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Summer Examinations 2013

M.Sc. in Applied Science (Bioinformatics with Systems Biology)

CS 6501 : Programming for Bioscientists I

Professor Ian Gent
Professor Barry O'Sullivan
Dr Joseph Manning

Answer all questions

$1\frac{1}{2}$ Hours

(*25%*) 1. a) Give an example in Python of a simple problem on sequences which can be solved *with or without indexing*, and one which can only be (easily) solved *with indexing*. The answer here should consist of a clear statement of both problems, two solutions for the first (with/without indexing), and one solution for the second. (*9%*)

b) Write both an *iterative* and a *recursive* version of a simple Python function to solve a given problem. The choice of problem is arbitrary, but it should be explained in the function comments. (*9%*)

c) Describe the main similarities and differences between *sequences* and *dictionaries* in Python, and state an advantage that each one has relative to the other. (*7%*)

(*25%*) 2. The *reverse complement* of a DNA sequence is obtained by interchanging each `A` and `T`, and each `C` and `G`, and then reversing the result. For example, the reverse complement of `AAGCTG` is `CAGCTT`.

Write a Python function `ReverseComplement( DNA )` to return the reverse complement of the DNA sequence `DNA`. Do *not* use the builtin Python function `reverse` here.

(recall that a DNA sequence is a string containing only the letters `A, C, G, T`)

(*25%*) 3. Write a Python function `Squeeze( filename )` to read in the file `filename` and write out each line of this file which differs from the previous line (its first line is always written); each output line should be prefixed with its line number in the *input* file.

Issue an appropriate error message if the file `filename` cannot be read.

(*25%*) 4. a) Write a Python function `Count( e, s )` to return the number of times that item `e` occurs in sequence `s`. (*8%*)

b) Write a Python function `Frequencies( s )` to return a dictionary, where each key is a distinct item in sequence `s` and the corresponding value is the number of times that the item occurs in `s`. For example:

```
Frequencies( "CCABBADCAC" )   ⇒  {"A":3, "C":4, "B":2, "D":1}
Frequencies( [4,7,4,7,4] )    ⇒  {4:3, 7:2}
```

(recall that the order in which keys appear in a dictionary is irrelevant) (*17%*)