COM2108: Functional Programming – Autumn 2019
Assignment 3: Breaking the Enigma Code

This assignment is 40% of the assessment for COM2108

This assignment is based on the 'Bombe Case Study', for which you have separate notes.

1 Tasks

1. Write Haskell code to implement the Bombe simulation following the design explained in the case study, assuming the simplifications made there: '*that the choice of rotors and reflector is always the same: referring to the assignment 2 spec the left rotor will be RI, the middle rotor RII and the right rotor RIII. The reflector is the standard one (which was called reflector B).*'

   You may make changes to the design, but **be sure to explain them if you do**. If your changes degrade the design, you will be penalised. You may make use of the code you submitted for assignments 1 and 2, or you may use the provided solution (which is on Blackboard).

2. Show that your code works for the examples given in `BombeTesting16.hs` on Blackboard. Be clear that you may not get perfect decodes: the menu may not provide enough constraints to deduce the whole stecker. You should be able to work out the message, however.

1 and 2 above count for 70% of the credit for this assignment. The remaining credit is reserved for experiments with the simulation.

3. Here are some suggestions. You don't have to follow all of them, and you are free to do your own thing provided you explain what it is.

   (a) How long does a menu need to be before the stecker based on it produces a correct decrypt? What happens if the menu is too short?

   (b) Does the answer to 3a depend on how many pairs there are in the stecker?

   (c) A given crib contains many menus. Amend the code so that several menus are used, rather than just the longest.

   (d) Remove the simplifications made in 1.

2 Marking Scheme

Task	Credit(%)
Correctness of:	
top level: search all offsets (breakEnigma, breakEA)	10
search for a Stecker given the offsets (findStecker)	10
follow implications from an initial Stecker (followMenu)	10
Bottom level: use of simulated enigmas and steckerAdd	10
(subtotal)	**40**
Extensions	**30**
Coding style	**15**
Documentation	**15**
Total	**100**

Table 1: Mark breakdown for assignment 3

For tasks 1 and 2 you should submit commented Haskell code, as one or more Haskell (`.hs`) files ready to run. Give your name in an initial comment of *every* Haskell file you submit. As before, make sure that your comments discuss any limitations (existing or addressed) related to your solution. If know *know* your code has limitations that you have been unable to address, you will achieve a better score if you clearly discuss those limitations in your comments than if you ignore them and your code fails test cases.

If you have completed the Extensions, you must include a short report of what you have done. In your report, you must do more than simply pose the questions and provide answers, you must *explain* how you came to your answers. For example, if you chose to answer 3b, it would not be enough to simply say "Yes" - you should explain why it is is dependent (and if possible, give the mathematical relationship). If the answer was "No," you should explain why it is *not* dependent.

You can only submit a single file so you should put all the files that you want to submit into a single directory, create an archive of that directory, and submit the archive. **MAKE SURE YOU CHECK THE CONTENTS OF YOUR ARCHIVE.** If you submit an empty or corrupted archive, you will receive a mark of 0 for this assignment.

3 Submission

Work should be submitted via the appropriate link on Blackboard. Any work received after the deadline will have standard lateness penalties applied (see `https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/late-submission`).

Once again, you are reminded that this is intended to be assessment of *your own, individual work*, in order to assess whether you have achieved with the learning outcomes for this module. For full details, see

`https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/unfair-means`