Your submission for this assignment ***must include your full name*** *(as it appears on cuLearn) and you nine-digit* ***student number as a comment*** *at the top of* ***every source file you submit****.*

Your submission for question 1 must be a ***single source file*** *with a* ***file name of either 'comp3007_f18_#########_a1_1.py' or 'comp3007_f18_#########_a1_1.java'*** *(with the number signs replaced by your nine-digit student number). It* ***must also be written*** *using either* ***Python 3*** *or* ***Java****.*

Your submission for questions 2 and 3 must be a ***single pdf file*** *with a* ***file name of 'comp3007_f18_#########_a1_2.pdf'*** *(with the number signs replaced by your nine-digit student number). It* ***must be written*** *using* ***Microsoft Word****,* ***Google Docs****, or* ***LaTeX****.* ***Photographs or scans*** *of handwritten submissions* ***will not be accepted*** *and will receive a mark of 0.*

Compress your submissions *into a zip archive named* ***'comp3007_f18_#########_a1.zip'***

Late assignments will not be accepted *and will* ***receive a mark of 0****.*

Submissions that crash *(i.e., terminate with an error) on execution will* ***receive a mark of 0****.*

The due date for this assignment is Saturday, September 22, 2018, by 11:00pm.

Question 1: "Camel Acrostics" (9.0 marks)

For the first question of this assignment, you will design and write a program – in either Python 3 or Java (your choice) – that uses several recursive algorithms to compute a property that I am calling the "Camel Acrostic" of a list.

The "Camel Acrostic" of a list of strings is the "word" that would be created if the elements of the list were written in "camel case" and then the capital letters were extracted and combined into a single string. (Please note that, for the purposes of this assignment, the term "camel case" refers to the capitalization of every word EXCEPT the first in the sequence.) As a clarifying example, the list of words:

```
["is", "tHis", "eXaMPLe", "SUFfIcIEntLY", "TELliNg"]
```

would be written in "camel case" as:

```
isThisExampleSufficientlyTelling
```

and would have a "Camel Acrostic" of:

```
TEST
```

The objective of this first question is to have you revisit the paradigm shift we discussed in the first lecture, where you as the designer focus your efforts, not upon describing "how" a result might be computed (as one might do in an imperative of procedural style), but instead upon "what" each of the necessary components is. Since we will (very shortly) be using recursive design exclusively, each of these definitions must be recursive in nature.

Your solution must meet the following requirements:

a) you must write a simple, non-recursive function that can change an uppercase letter into a lowercase one (and if you use an unnecessarily complex branching structure to do this, I will be very sad and you will be harshly penalized),

b) you must write a recursive function that will change the "case" of an argument string (written in "studly caps" i.e., with random capitalization) such that the return value is an equivalent string with ONLY the first letter capitalized; this function MUST use the function you wrote for requirement (a) above,

c) you must write a recursive function that will take a list of strings as an argument and create a single string in "camel case" (i.e., with an uppercase letter at the beginning of every word EXCEPT the first) as the return value; this function MUST use the function you wrote for requirement (b) above,

d) you must write a recursive function that will take a string as an argument and retain and combine only the capitalized letters into a new string return value; this function MUST use the function you wrote for requirement (c) above, and

e) you must write a simple interface for the user (used for testing your submission); this interface must permit the user to type a string, parse that string into a list of words, compute the "Camel Acrostic" (using the function you wrote for requirement (d) above), display the resulting "word", and finally ask the user if they would like to continue (looping back to the beginning or terminating the program as required).

Please note that you must write your own function for requirement (a); you cannot use a built-in function. Please also note that the looping control structure used for the interface mentioned in required (e) above must be the only looping control structure used in your entire program – the functions described in requirements (b), (c), and (d) must be recursive and cannot use looping control structures.

This program (along with any other program submitted in this class) must be a completely original work, authored by you and you alone, prepared for this offering (i.e., Fall 2018) of COMP3007. Do not discuss this (or any other) question with anyone except the instructor or the teaching assistants, and do not copy materials from the internet or any other source.

The exercises you complete for the second and third question of this assignment must also be completed individually and will be different depending on your student number. Please ensure you have selected the correct option before submitting, or else you may receive a mark of zero for these questions.

Question 2: "Beta Reduction" (3.5 marks)

If the LAST DIGIT of your student number is 1, 2, or 3, then use the following expression:

$$\Big(\lambda a.\lambda b.(\lambda c.\lambda d.\lambda e.c\ d\ e)\ a\ b\ \big(\lambda f.(\lambda g.f)\big)\Big)\ \big(\lambda h.(\lambda i.h)\big)\ \big(\lambda j.(\lambda k.j)\big)$$

If the LAST DIGIT of your student number is 4, 5, or 6, then use the following expression:

$$\Big(\lambda a.\lambda b.(\lambda c.\lambda d.\lambda e.c\ d\ e)\ a\ b\ \big(\lambda f.(\lambda g.f)\big)\Big)\ \big(\lambda h.(\lambda i.h)\big)\ \big(\lambda j.(\lambda k.k)\big)$$

If the LAST DIGIT of your student number is 7, 8, 9, or 0, then use the following expression:

$$\Big(\lambda a.\lambda b.(\lambda c.\lambda d.\lambda e.c\ d\ e)\ a\ b\ \big(\lambda f.(\lambda g.f)\big)\Big)\ \big(\lambda h.(\lambda i.i)\big)\ \big(\lambda j.(\lambda k.j)\big)$$

For this question, you must perform a β-reduction on whichever of the above expressions corresponds to the LAST DIGIT of your student number. Each of these is VERY SIMILAR to some of the in-class examples that can be found in the lecture notes entitled "λ-Calculus as a Programming Language", starting on slide 31.

Not counting the initial expression, each of the above can be β-reduced in exactly 7 steps. SHOW EVERY STEP. You will know you have finished when your expression is reduced to something of the form $\big(\lambda\blacksquare.(\lambda\blacksquare.\blacksquare)\big)$, at which point you will not be able to reduce the expression further.

Question 3: "Lazy Evaluation" (2.5 marks)

For this question, you will consider the following collection of Haskell function definitions and then evaluate whichever of the expressions (of the following page) corresponds to the 2nd LAST DIGIT of your student number.

```
puzzle :: Int -> Int -> Int
puzzle a b = a * b

enigma :: Int -> Int -> Int
enigma a b = a - b

secret :: Int -> Int -> Int
secret a b = b - a
```

If the 2nd LAST DIGIT of your student number is 1, 2, or 3, then trace the evaluation of:

```
enigma (secret (puzzle 1 3) 5) 7
```

If the 2nd LAST DIGIT of your student number is 4, 5, or 6, then trace the evaluation of:

```
puzzle (enigma (secret 2 4) 6) 8
```

If the 2nd LAST DIGIT of your student number is 7, 8, 9, or 0, then trace the evaluation of:

```
secret (puzzle (enigma 3 5) 7) 9
```

To receive full marks for this question you must SHOW (in the correct order) EVERY REDUCIBLE EXPRESSION REPLACEMENT performed by Haskell during the evaluation. You may wish to review the lecture notes entitled "Equational Reasoning in Haskell" before attempting to do so.

n.b., Not counting the initial expression, each of the above expressions can be reduced to an integer final answer using exactly 6 steps. This includes the steps required for evaluating the arithmetic expression. SHOW EVERY STEP and do not hesitate to create and execute the program from the previous page in order to help you confirm your final answer.