Computational Thinking and Problem Solving (COMP1002) and Problem Solving Methodology in Information Technology (COMP1001)

Assignment 3
(Due on **3 December 2021 (Fri) at 12:00 noon**)

1. [20 marks] We consider a coin-changing problem. For a currency with coins c_1, c_2, ..., c_N, the problem is to find the minimum number of coins needed to make x dollars of change, as well as how many coins are needed for each of c_i where $1 \leq i \leq N$, assuming there are always enough coins. For example, in Hong Kong, we have 0.1-dollar, 0.2-dollar, 0.5-dollar, 1-dollar, 2-dollar, 5-dollar and 10-dollar coins. For x = 63.7, the minimum number of coins is 10: 6 x 10-dollar coins, 1 x 2-dollar coin, 1 x 1-dollar coin, 1 x 0.5-dollar coin and 1 x 0.2 dollar coin. You can assume x must be a sum of the face value of coins. Write the pseudocode to solve this problem.

2. [30 marks] Suppose there is a number game between two persons, Alice and Bob. At the beginning of the game, each player starts with a number 0. Each player takes turns and calls a number which is either 1 or 2 bigger than his/her current value. At any moment, the difference between the two numbers of Alice and Bob should not be greater than 2. The winner is the one who calls 5 first.

   For example, at the beginning, Alice holds 0 and Bob holds 0. Alice calls 1. Bob calls 1. Alice calls 3. Bob calls 3. Alice calls 5. Alice wins the game.

   Another example: Alice calls 2. Bob calls 1. Alice cannot call 4 because her number would be 3 greater than Bob. So, her only choice is to call 3. Bob calls 3. Alice calls 5. Alice wins the game.

   Assume Alice will always take the first call.

   Answer the following questions:

   a) Define the "state" of this game. Draw a graph to represent all states and their relationship.

   b) Can Bob win the game? Why? State your assumption(s).

3. [15 marks] Create your own *split* function in Python, using the name *mySplit*, which returns a list of words based on two parameters, an input string, and a list of separators such as `[‵,′, ′?′, ′.′]` flexibly defined by you. The punctuations and symbols are not considered as a part of a word.

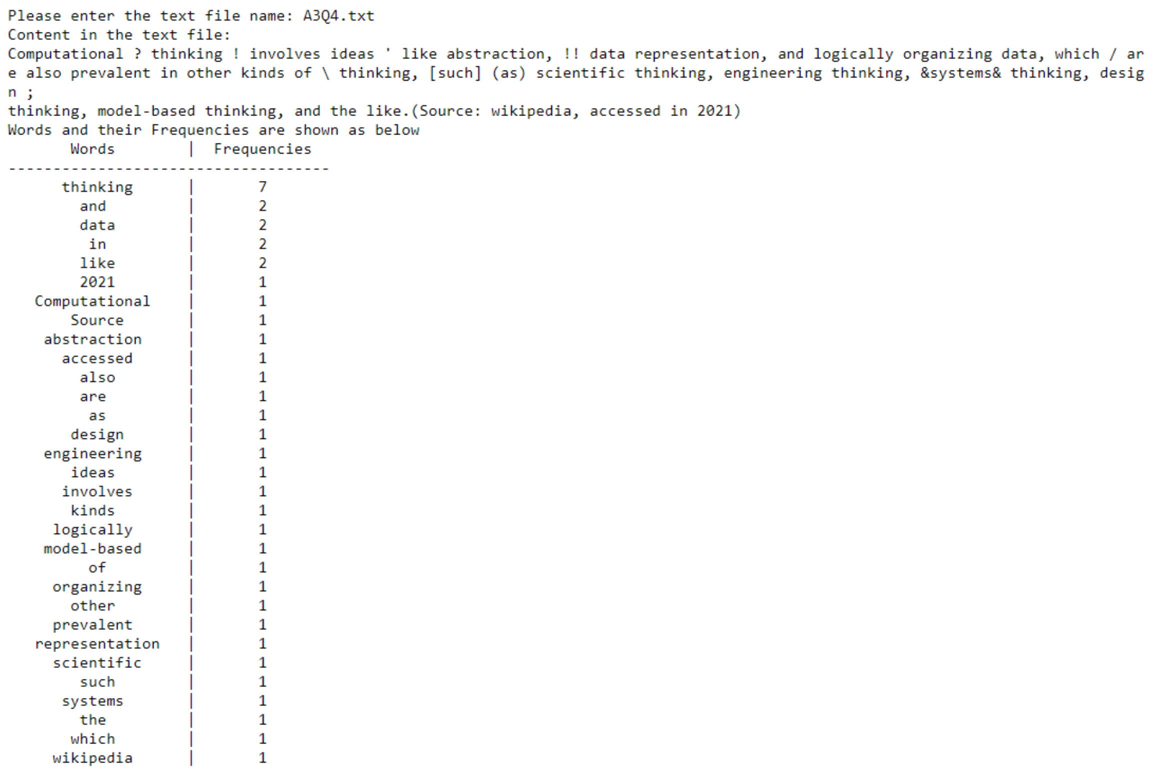
   Write a function called *a3q3()* to test *mySplit*. When *a3q3()* is called, the input/output of your program will look like below:

```
Please input a paragraph: a for apple! b for boy; and c for cat, d for dog. but e for egg? No grammar mistakes checking!
A list of words from your input:
 ['a', 'for', 'apple', 'b', 'for', 'boy', 'and', 'c', 'for', 'cat', 'd', 'for', 'dog', 'but', 'e', 'for', 'egg', 'No', 'gramma
r', 'mistakes', 'checking']
```

   Zero mark will be awarded if any built-in/external functions, other than the function, *append()*, of list, are used in *mySplit*.

4. [35 marks] Create a Python function, named *wordFeq*(), which returns a sorted list of words or numbers based on frequencies from input paragraph(s). If more than one words are having the same frequency, the sorted results are determined by number, alphabetic case and then alphabetic order (i.e., the order in ASCII code). Your program is required to call *mySplit* developed in Q3. The words are case-sensitive.

   Write a function called *a3q4()* to test your *wordFeq*. When *a3q4()* is called, it should read in a text file which contains the input paragraph(s). The input/output of your program will look like below:



```
Please enter the text file name: A3Q4.txt
Content in the text file:
Computational ? thinking ! involves ideas ' like abstraction, !! data representation, and logically organizing data, which / ar
e also prevalent in other kinds of \ thinking, [such] (as) scientific thinking, engineering thinking, &systems& thinking, desig
n ;
thinking, model-based thinking, and the like.(Source: wikipedia, accessed in 2021)
Words and their Frequencies are shown as below
       Words        |  Frequencies
-----------------------------------
      thinking      |       7
        and         |       2
        data        |       2
         in         |       2
        like        |       2
        2021        |       1
   Computational    |       1
       Source       |       1
    abstraction     |       1
      accessed      |       1
        also        |       1
        are         |       1
         as         |       1
       design       |       1
    engineering     |       1
       ideas        |       1
      involves      |       1
       kinds        |       1
     logically      |       1
    model-based     |       1
         of         |       1
     organizing     |       1
       other        |       1
     prevalent      |       1
   representation   |       1
     scientific     |       1
        such        |       1
      systems       |       1
        the         |       1
       which        |       1
     wikipedia      |       1
```

Submission Instructions

Follow the steps below:

1. Create a folder and name it as <student no>_<your name>,
   e.g., `12345678d_CHANTaiMan`
2. For Q1 and Q2, type/draw your answers in a word document and save it as a `.pdf` file. Name the single `.pdf` file as A3_<student no>_<your name>`.pdf`,
   e.g., `A3_12345678d_CHANTaiMan.pdf`
3. For Q3 and Q4, submit the source file (`.py`). Name the `.py` files as A3_Q<question no>_<student no>_<your name>`.py`,
   e.g., `A3_Q3_12345678d_CHANTaiMan.py`
4. Put all the `.pdf` and `.py` files into the folder created in Step 1.
5. Compress the folder (`.zip`, `.7z`, or `.rar`).
6. Submit the file to Blackboard.

A maximum of **3 attempts** for submission are allowed. **Only the last attempt will be graded**. A late penalty of 5% per hour will be imposed.

Any wrong file naming and submission will be given ZERO mark. If you are using Windows, the file extension may be hidden by the operating system. Follow the steps of below links to make sure the file extension is not hidden:

https://www.howtohaven.com/system/show-file-extensions-in-windows-explorer.shtml

If your program cannot be run successfully (i.e., having any syntax error(s)), ZERO mark will be awarded for that program, regardless of how much you have coded.

This assignment is an individual work. All work must be done on your own. Plagiarism is serious offence. You are not allowed to consult any external channels, e.g., discussion forums, and copy code from any web resources, to assist your completion of your assignments. The Moss (https://theory.stanford.edu/~aiken/moss/) system will be adopted for plagiarism checking for program code. Submissions with high similarity, in terms of code patterns and structures, in addition to direct-copy-and-paste, will be extracted and reviewed. Any plagiarism cases (both copier and copiee) will be given ZERO mark plus a deduction of the maximum mark of this assignment. Serious cases would be submitted to the Student Discipline Task Group (SDTG) of the department for further disciplinary actions.