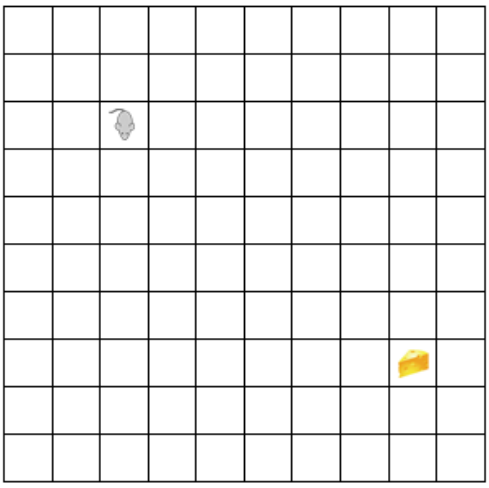
Computational Thinking and Problem Solving (COMP1002)

Assignment 2

(Due on **28 October 2022 (Fri) at 12:00 noon**)

1. [20 marks] Suppose that is a mouse, which can freely move on a 10 x 10 2D plane. It initially occupies one of the slots of the plane. Its goal is to find out the cheese located in another slot. Every time, it can only move to its neighbor slot in either of the directions, N, E, S or W (its head in the diagram is facing S). It cannot move across the boundary of the plane. The mouse is shorted-sighted that it can only see objects that are up to 2 slots away horizontally and vertically and 1 slot away diagonally (not including its current slot). If it can see the cheese, it will move in a path that leads to the cheese. Otherwise, it will continue to move to next slot randomly.



Answer the following questions:

a) Write down the pseudocode of a procedure, called `move()`, that contains the logic of a single mouse move. [5 marks]

b) Write down the pseudocode of a function, called `findCheese()`, that contains the logic of checking the existence of the cheese in the visible area of the mouse. It should return true or false, and the location of the cheese, relative to the mouse. [5 marks]

c) Write down the pseudocode of a procedure, called main(), that contains the logic for the mouse to achieve its goal. You must use `move()` and `findCheese()` in `main()`. [10 marks]

2. [25 marks] In this course, you have learnt the concepts of binary addition. Suppose you are given an integer, `A`, in 2s-complement binary representation with arbitrary length.

Answer the following questions:

a) Given `A` of length `m`. Write down the pseudocode, in terms of a function, to convert it to length `n`, where `n > m`. [10 marks]

b) Use your answer in a). Given `A` of length `m` and `B`, another 2s-complement integer, of length `n`, write down the pseudocode, in terms of a function, to perform addition of `A` and `B`. You must demonstrate the addition in bit-by-bit level. Note that you may have to extend the resulting number to maintain a correct representation. [15 marks]

3. [30 marks] Create a Python program with the following requirements:

   a) **Create** your own *max* function, called *myMax*, which finds the maximal number and its *i*-based location (*i* starting from 1) in a series of different numbers (no two or more numbers are of the same value and there are at least two numbers). The *while* loop is required to build the *myMax* function. Use *docstring* to describe your function. Marks will be deducted if *for* loop and/or the built-in function for *max* are used. In the demonstration of calling your *myMax* function, print the *docstring* about the function, **and** show how to use your function as below:

```
Please enter a list of different numbers separated by ',': 1,-3,4.5,5,18,-1,3,-4
The maximal number is 18 .
Its location index is 5 .
```

   [15 marks]

   b) Using *myMax* function created in b), **create** your sorting function, called *mySort*, to sort a set of different numbers. The function will return a list of sorting values in descending order. The *while* loop statement is required. Use *docstring* to describe your function. Marks will be deducted if the built-in functions for *sorting* are used. If you use the other sorting method without calling your *myMax* function, only a maximum of half of the marks of this question will be awarded. You may use build-in functions to add or remove an element from a list. In the demonstration of calling your sorting function, print the *docstring* about the function, **and** show how to use your function as below:

```
Please enter a list of different numbers separated by ',': 1,-3,4.5,5,18,-1,3,-4
A list of sorting values in descending  order: [18, 5, 4.5, 3, 1, -1, -3, -4] .
```

   [15 marks]

4. [20 marks] Develop a function, on input of a string, *s*, and a character, *c*, returns the number of occurrence of *c* in *s*, and a list of index(s) (one-based) of *s* that represent(s) the location(s) of *c*. Use *for* loop in your implementation. Use *docstring* to describe your function. Marks will be deducted if *while* loop and any built-in functions are used in the implementation of this function. In the demonstration of calling your function, print the *docstring* about the function, **and** show how to use your function as below:

```
Input text: Never Too late To Start. Please go ahead with us.
Input a character to be searched: t
The character t in the text occured 4 times at [13, 20, 23, 44].
```

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 your answers in a word document and save it as a **`.pdf`** file. Name the single **`.pdf`** file as A2_<student no>_<your name>**`.pdf`**,
   e.g., **`A1_12345678d_CHANTaiMan.pdf`**
3. For Q3, and Q4, submit the source file (**`.py`**). Name the **`.py`** files as A2_Q<question no>_<student no>_<your name>**`.py`**,
   e.g., **`A2_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. It is your obligation to check carefully the files in your submission.

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)) when it is triggered, 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.