ANGLO-CHINESE JUNIOR COLLEGE
JC1 PROMOTIONAL EXAMINATION

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **26 August 2021**

1 hour 30 minutes

Additional Materials: Electronic version of `Task1_template.txt` data file
Electronic version of `Sums.txt` data file
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** questions.

All tasks must be done in the computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Approved calculators are allowed.

Save each task as it is completed.

The use of built-in functions, where appropriate, is allowed for this paper unless stated otherwise.

Note that up to **3** marks out of 50 will be awarded for the use of common coding standards for programming style.

The number of marks is given in brackets [ ] at the end of each question or part question.
The total number of marks for this paper is 50.

Instruction to candidates:

Your program code and output for each of Task 1 and 2 should be downloaded in a single .ipynb file. For example, your program code and output for Task 1 should be downloaded as
`TASK1_<your name>_<index number>.ipynb`

1 While Binary Search Trees (BST) are commonly used to store integers, they can also be used to store other data types. A computing student would like to store strings in the BST.

Task 1.1

It is suggested that the value of a string `s` can be calculated by the following pseudocode .

```
total ← 0
FOR i ← 0 to length(s) - 1
    total ← total + ASCII(s[i])
ENDFOR
RETURN total
```

Write a function `string_value(s)` that:

- takes a string `s`;
- calculates the value of that string using the pseudocode;
- returns that value. [2]

Task 1.2

The file `Task1_template.txt` contains code for the `Node`, `Queue` and `BST` classes. Copy and paste the provided code into Jupyter Notebook.

Extend the BST class with the following methods:

- `insert(s)` inserts the string `s` into the BST.
  If `s` has the same value as an existing node, `s` should be discarded.
- `in_order_list()` returns a list of the strings in the BST, using an in-order traversal. [8]

Task 1.3

Level-order traversal is another form of traversal where the nodes of the BST are visited level by level. The algorithm to perform level-order traversal is described below:

1. Check if the tree is empty. If the tree is empty, display `'Empty Tree!'`.
2. Create a Queue.
3. Enqueue the root of the BST into the Queue.
4. Iterate through the Queue while it is not empty and perform the following:
   a. Remove the first node in the Queue and display its data
   b. Enqueue the removed node's left child into the Queue
   c. Enqueue the removed node's right child into the Queue

Extend your `BST` class with the `level_order_traversal()` method using the algorithm described. [5]

Task 1.4

Write a function `random_string(n)` that:

- Takes an integer `n`;
- Randomly uses only uppercase or lowercase letters to create a string of length `n`;
- Returns that string.

All letters have an equal probability of appearing in the string, and the same letter may appear more than once within the string.

Example:
`random_string(4)` may return `'AAzD'`. [3]

Task 1.5

Write code to perform the following:

- Create a list named `Lst_Of_Strings` and append randomly generated strings of letters to it. There should be one string of length one, two strings of length two, three strings of length three, and four strings of length four in the list.
- Create a BST named `Tree` and insert all strings in `Lst_Of_Strings` into `Tree`.
- Display the strings using a level-order traversal. [5]

Download your program code and output for Task 1 as
`TASK1_<your name>_<index number>.ipynb`

2 A string of digits, mathematical operators (`'+'` and `'-'`) and brackets (`'('` and `')'`) may represent a mathematical expression that can be evaluated to give an integer.

You are not allowed to use the `eval` function for this Task.

Task 2.1

Write a function `calculate_1(s)` that takes a string of digits and `'+'`s, evaluates the mathematical expression, and returns the answer as an integer.

There will be no `'-'` or brackets in the expression. You do not need to check the expression for validity.

Example:
`calculate_1('123+456')` returns `579`. [5]

Task 2.2

Modify the function `calculate_2(s)` to write a new function `calculate_2(s)` that takes a string of digits, `'+'`s and `'-'`s, evaluates the mathematical expression, and returns the answer as an integer.

There will be no brackets in the expression. You do not need to check the expression for validity.

Examples:
`calculate_2('123-456+789')` returns `456`.
`calculate_2('-123+456+789')` returns `1122`. [4]

Task 2.3

Write a function `calculate(s)` that takes a string of digits, mathematical operators and brackets, evaluates the mathematical expression, and returns the answer as an integer. You do not need to check the expression for validity.

Examples:
`calculate('123-(456+789)')` returns `-1122`.
`calculate('12-(34-56)+78')` returns `112`.
`calculate('(12-34)+(56-78)')` returns `-44`.
`calculate(' 1+(2-(3+4)+5)-6')` returns `-5`. [10]

Task 2.4

The file `Sums.txt` contains some mathematical expressions.

Write program code to evaluate each sum and write the answers to a new file, `Answers.txt`.

The answers should be given in the same order as the corresponding sums, with each answer on a new line. [5]

Download your program code and output for Task 2 as
`TASK2_<your name>_<index number>.ipynb`