HWA CHONG INSTITUTION
C2 BLOCK TEST 2021

COMPUTING
Higher 2

28 June 2021 **Paper 2 (9569 / 02)** **1400 -- 1530 hrs**

Additional Materials:

Electronic version of `WORDS.txt` data file

Electronic version of `STAFF.json` data file

Electronic version of `TEXT.txt` data file

READ THESE INSTRUCTIONS FIRST

Answer **all** questions.

The maximum mark for this paper is **45**.

The number of marks is given in brackets [ ] at the end of each question or part question.

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.

This document consists of **6** printed pages.

1. Design the server program and client program for an asymmetric game to guess a 5-letter word.

   The file `WORDS.txt` contains words of 5 distinct letters. The server generates a random word using this file. The client has 5 chances of guessing whether a particular letter is in the guessed word. After all the 5 guesses, the client will make the final guess of the word and get the outcome from the server. Below is a sample run of the client.

```
Guess a word of 5 distinct letters!

Guess a letter in the word:a
YES
Guess a letter in the word:e
YES
Guess a letter in the word:d
NO
Guess a letter in the word:h
NO
Guess a letter in the word:m
YES

Guess the word:meant

You LOSE! The correct word is anime
```

   You do not need to handle invalid input of client. [9]

   Save your program code as
   `TASK1_client_<your name>_<ct>.py` and
   `TASK1_server_<your name>_<ct>.py`

   Run the program. Produce a screenshot of the output and save it as

   `TASK1_<your name>_<ct>.jpg`

2. The task is to create a NoSQL database for a company to manage its staff records on the recent vaccine. The vaccine requires two doses to be completed.

Task 2.1

Write a python program to:

- Create a MongoDB database named `Vaccine` and a new collection named `Staff`.
- Insert the documents into the `Staff` collection using the data file `STAFF.json`.

Save your program code as `TASK2_1_<your name>_<ct>.py` [2]

Task 2.2

Write program code which make use of `Staff` collection in `Vaccine` database to:

- display the `Name`, `Age` and `Reason` for staff who do not accept to take the vaccine;
- display the number of staff with at least one dose of vaccine taken;
- add Charlie Lee's second dose on "5 Apr 2021" and display his updated record;
- display the name(s) of staff with at least 50 years old and completes both doses

All outputs should have appropriate messages to indicate what you are showing.

Save your program code as `TASK2_2_<your name>_<ct>.py`

Produce a screenshot of the output and save it as
`TASK2_2_<your name>_<ct>.jpg` [9]

3. Name your Jupyter Notebook as

`TASK3_<your name>_<ct>.ipynb`

The task is to implement a Binary Search Tree using Object Oriented Programming.

For each of the sub-tasks, add a comment statement, at the beginning of the code using the hash symbol '`#`' to indicate the sub-task the program code belongs to, for example :

```
In [1]:  # Task 3.1
         Program Code

In [2]:  # Task 3.2
         Program Code

In [3]:  # Task 3.3
         Program Code
         output:

In [4]:  # Task 3.4
         Program Code
         output:
```

Task 3.1

Implement class `TreeNode` with the following attributes and their get and set methods.

- Attributes:
  - `word` – the node's value for a word
  - `left_ptr` – the left pointer of the node
  - `right_ptr` – the right pointer of the node

[3]

Task 3.2

Implement class `BinarySearchTree` with the following **three** methods.

- `add(word)`
  - adds the unique word into the Binary Search Tree

- `inOrder()`
  - performs an in-order tree traversal
  - displays the words in the in-order traversal order
  - returns an array of tree nodes, in the in-order traversal order

- `preOrder()`
  - performs a pre-order tree traversal
  - displays the words in the pre-order traversal order

[13]

Task 3.3

`TEXT.txt` is a text file containing unique words.

Write program code to:

- create a Binary Search Tree data structure using the classes created in Task 3.1 and Task 3.2
- read the words from file `TEXT.txt`
- use the `add()` method to add the words into the Binary Search Tree.
- call the `inOrder()` method to display the words stored in the tree.

[4]

Task 3.4

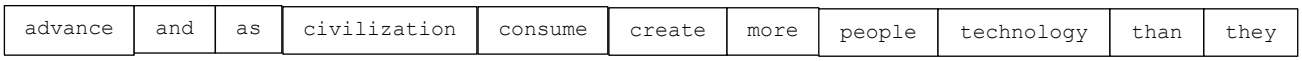
One way to balance a Binary Search Tree is to:

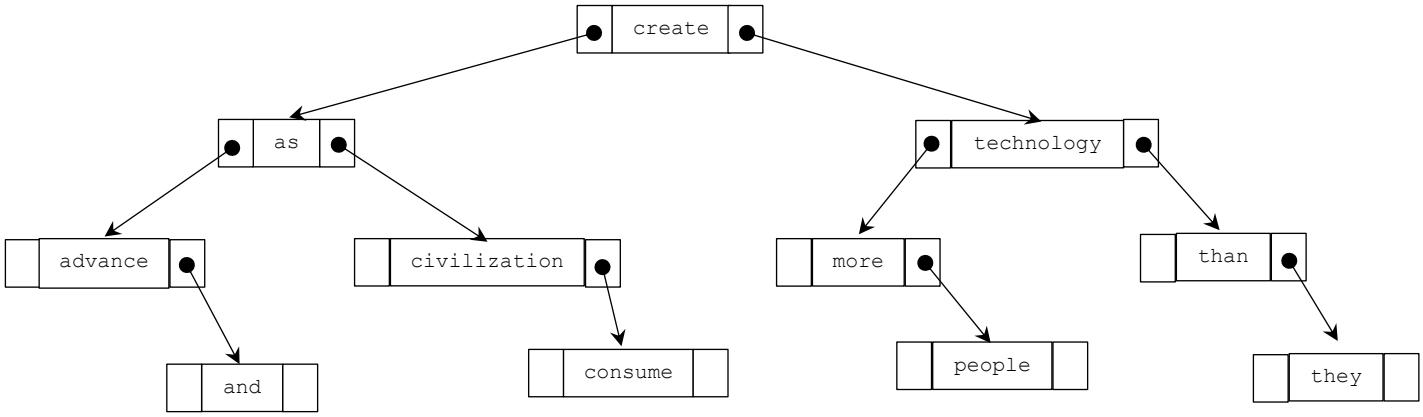
- store all tree nodes to an array in sorted order

- reconstruct a new Binary Search Tree by
  (a) getting the middle of the array as the root node

  (b) recursively do the same for the left half and right half of the array
    - get the middle of the left half as the left child of the root node created in step **(a)**
    - get the middle of the right half as the right child of the root node created in step **(a)**

The following diagrams show a sorted array of tree nodes and the created balanced Binary Search Tree.





Write program code to:

- create a balanced Binary Search Tree using the sorted array returned from the `inOrder()` method in Task 3.3.

- call the `preOrder`() method to display the words stored in the created balanced Binary Search Tree.

[5]

Download your program code and output for Task 3 as
`TASK3_<your name>_<ct>.ipynb`