Name:		Index Number:		Class:	

DUNMAN HIGH SCHOOL
Mid-Year Examination
Year 6

COMPUTING (Higher 2)

9569/02

Paper 2 (Lab-based)

28 June 2023

3 hours

Additional Materials: Insert
Electronic version of `CARPLATES.txt` file
Electronic version of `Task2data.py` file
Electronic version of `STUDENTS.csv` file
Electronic version of `FOLLOWING.csv` file
Electronic version of `USERS.csv` file

READ THESE INSTRUCTIONS FIRST

Write your name, index number and class on the work you hand in.
Write in dark blue or black pen on both sides of the paper.
You may use an HB pencil for any diagrams or graphs.
Do not use staples, paper clips, glue or correction fluid.

Answer **all** the 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 **4** marks out of 100 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 100.

Instructions to candidates:

Your program code and output for each of Task 1 to 4.4 (except Task 4.3) should be saved in a single `.ipynb` file using Jupyter Notebook. For example, your program code and output for Task 1 should be saved as:

`TASK1_<your name>_<centre number>_<index number>.ipynb`

Make sure that each of your `.ipynb` files shows the required output in Jupyter Notebook.

1 Name your Jupyter Notebook as:

`TASK1_<your name>_<centre number>_<index number>.ipynb`

Vehicle registration plates in Singapore are administered by the Land Transport Authority. All vehicles in Singapore are required to display front and back plates bearing its registration number.

A typical vehicle registration number comes in the format “SXX #### Y”, where:

S – Vehicle class ("S" stands for a private vehicle since 1984)
X – Alphabetical series ("I" and "O" are not used to avoid confusion with "1" and "0")
– Numerical series (from 1 to 9999)
Y – Checksum letter

The calculation of the checksum letter is as follows:

Example: SBA 1234

1) ignore the letter "S"
2) B = 2, A = 1 (A=1, B=2, C=3 and so on)
3) SBA 1234 is converted to 21 1234 (if there are less than 4 numbers in the Numerical Series, leading 0s are added to form the 4 numbers. E.g., SCD 45 will be converted to 34 0045)
4) Each individual number is then multiplied by 6 fixed numbers (9, 4, 5, 4, 3, 2) based on the position of the number, E.g., for SBA 1234 the calculation will be 2 x 9, 1 x 4, 1 x 5, 2 x 4, 3 x 3, 4 x 2
5) add the numbers together, 18 + 4 + 5 + 8 + 9 + 8 = 52
6) 52 divided by 19 = 2 with remainder 14
7) match the remainder with these 19 letters, 0=A, 1=Z, 2=Y, 3=X, 4=U, 5=T, 6=S, 7=R, 8=P, 9=M, 10=L, 11=K, 12=J, 13=H, 14=G, 15=E, 16=D, 17=C, 18=B
8) remainder 14 equal to letter G
9) hence the full vehicle registration number is SBA 1234G

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 [ ]: #Task 1.1
        Program code
```

Output:

Task 1.1
Write a function `task1_1(filename)` that:

- takes a string `filename` which represents the name of a text file
- reads in the contents of the text file
- returns the content as a list of strings. (Note: '\n' must be removed before adding into the list [3]

Call your function `task1_1` with the file `CARPLATES.txt`, printing the returned list and its length, using the following statements:

```
result = task1_1("CARPLATES.txt")
print(result)
print(len(result))
```
[2]

Task 1.2
One method of sorting is the quick sort.

Write a function `task1_2(list_of_strings)` that:

- takes a list of strings (vehicle registration number)
- implements a quick sort algorithm
- returns the list of vehicle registration number from oldest to youngest (In 1984, vehicle registration number started with "SBA", followed by "SBB", "SBC" and so on. The latest vehicle registration number today is "SNK") [7]

Call your function `task1_2` with the contents of the file `CARPLATES.txt`, printing the returned list, for example, using the following statement:

```
print(task1_2(task1_1("CARPLATES.txt")))
```
[1]

Task 1.3
Write a function `task1_3(input_value)` that returns a string:

The function should:

- validate that the parameter's (`input_value`) first character is `"S"` followed by 2 other alphabets
- validate that the numbers range from 1 to 9999
- return `"error"` if the value received is invalid for any reason
- calculate the checksum letter for valid vehicle registration number
- return the calculated checksum letter for valid vehicle registration number. [6]

Task 1.4
Write a function `task1_4(input_filename, valid_filename, invalid_filename)` that:

- accepts three parameters:
  - `input_filename` represents the input file name
  - `valid_filename` represents the output file name that stores valid vehicle registration numbers
  - `invalid_filename` represents the output file name that stores the invalid vehicle registration numbers
- uses your `task1_1` to read from the input file
- uses your `task1_3` to calculate the checksum letter for each vehicle registration

number
- concatenate the checksum letter to the back of valid vehicle registration numbers and write the completed vehicle registration number into the valid output file (one vehicle registration number per line)
- write the invalid vehicle registration number into the invalid output file (one vehicle registration number per line) [5]

The function should read the vehicle registration numbers from `CARPLATES.txt` and write into one of the below files accordingly:

```
VALID_<your name>_<centre number>_<index number>.txt
INVALID_<your name>_<centre number>_<index number>.txt
```

Run the program. [1]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as:

`TASK2_<your name>_<centre number>_<index number>.ipynb`

A shop owner wishes to keep track of the names of the items she has in her store. She decided to write a class, `Queue`, of fixed length 10, to store the values.

The pseudocode of the constructor is as given below:

```
MAX_SIZE = 10
ARRAY queue[max_size]
front = 0
rear = -1
```

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 [ ]: #Task 2.1
        Program code

        Output:
```

Task 2.1

Write the `Queue` class in Python. Include the following methods:

- `is_full()` returns a Boolean value: `True` if the queue is full; `False` if the queue is not full
- `is_empty()` returns a Boolean value: `True` if the queue is empty; `False` if the queue is not empty
- `enqueue(item_name)` will insert the `item_name` to the end of the queue
- `dequeue()` will return the first element in the queue. If the queue is empty, return `None`.
- `display()` should output a string showing all the elements in the queue in order followed by the value of the front index pointer and rear index pointer. For example,

```
Juice Milk Chips Eggs None None None None None None
front:  0 rear:  3
```

Copy and paste the code in the file `Task2data.py` into your Jupyter Notebook. Execute the code to create and populate the `Queue`. [8]

Task 2.2

The shop owner realizes that the items in the `Queue` are not sorted in alphabetical order. Hence, she decided to implement a `Binary Search Tree` instead to store the names of the items in the store.

The tree is implemented using Object-Oriented Programming (OOP).

The class `Tree` contains three properties:

- `left_pointer` points to the left subtree
- `right_pointer` points to the right subtree
- `data` is the name of the item

The class `Tree` contains the following methods

- a constructor to set the `left pointer` and `right pointer` to `None`, and the `data` to its parameter
- a recursive method to take the parameter and insert it in the correct position in the `tree`
- a recursive method to use in-order traversal to output the data in the `tree`.

Write the `Tree` class in Python. [8]

Write the main program to:

- declare a new instance of `Tree`
- insert the names of the store items into `Tree` by dequeuing from `Queue` in Task 2.1
- display the names of the store items in alphabetical order using the in-order traversal of `Tree`. [3]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as:

`TASK3_<your name>_<centre number>_<index number>.ipynb`

A school wants to use Python Programming and object-oriented programming to store information about its students.

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 [ ]: #Task 3.1
        Program code

        Output:
```

Task 3.1
The `Student` class has the following private data fields:

- `studentid` – stored as a integer
- `full_name` – stored as a string
- `gender` – stored as a string, for example, Male, Female
- `date_of_birth` - initialized with a string with the format YYYY-MM-DD
- `subject_combination` – stored as a List of subject titles the student is taking

The class contains all the appropriate methods to set and access the above private data fields. It also includes one additional method:

- `show_details( )`
  - display `studentid`, `full_name`, `gender`, `date_of_birth` and `subject_combination` in separate lines. Example:

```
Student ID: 5
Full Name: Oon Yi Min
Gender: Female
Date of Birth: 2005-07-23
Subject_Combination:
Subject 1: H2 Biology
Subject 2: H2 Chemistry
Subject 3: H2 Mathematics
Subject 4: H2 Economics
```

Write program code in Python to define the class `Student`. [8]

Task 3.2
The csv file, `STUDENTS.csv`, contains information of a number of students. Each row is a comma-separated list of data of the following:

- Student ID
- Full Name
- Gender
- Date of Birth in the form YYYY-MM-DD
- Subjects student is taking (4 to 5 subjects per student)

Write program code to read in the information from the csv file, creating an instance of the `Student` class for each student. Store all instances into a global students list named `students`.

Write program code in Python to show the details of all students in the `students` list. [6]

Task 3.3

Write program code to declare a recursive function to:

- Implement a recursive binary search on the `students` list from Task 3.2 based on Student ID (`Student` list is already sorted by Student ID)
- Return the position of the student in the list
- Return -1 if student is not in the list

Write a program to:

- Read in an integer value (Student ID) from the user
- Call the binary search function with the integer input (Student ID)
- Display the details of the student if the student is in the list
- Output "`Student Not Found`" if student is not in the list
- Ask the user if they would like to search again. Repeat the process if User input "`Yes`" [9]

Test your program by running the application with the Student ID 15 followed by 105. [1]

Save your Jupyter Notebook for Task 3.

4 Name your Jupyter Notebook as:

`TASK4_<your name>_<centre number>_<index number>.ipynb`

You are the founder of a new social media platform, Photogram. In Photogram, each user can only create one account and they can follow other users. The database will have two tables: a table to store the users' information and a table to store the User IDs of other people each user is following.

`User (UserID, Name, Gender, Country)`
`Following (UserID, Following_UserID)`

`User:`

- `UserID` – unique user account number, for example, 12
- `Name` – the name of the user
- `Gender` – the gender of the user, for example, Male, Female
- `Country` – the country of origin of the user, for example, Singapore

`Following:`

- `UserID` – unique user account number, for example, 12
- `Following_UserID` – unique user account number, for example, 12

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 [ ]: #Task 4.1
        Program code
```

`Output:`

Task 4.1
Write a Python program that uses SQL code to create the database `photogram.db` with the two tables given. Define the primary and foreign keys for each table. [5]

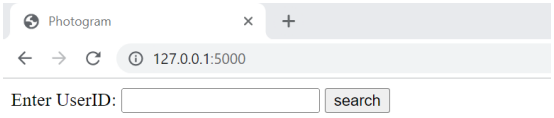
Task 4.2
The files `USERS.csv` and `FOLLOWING.csv` store comma-separated values for `User` and `Following` tables respectively.

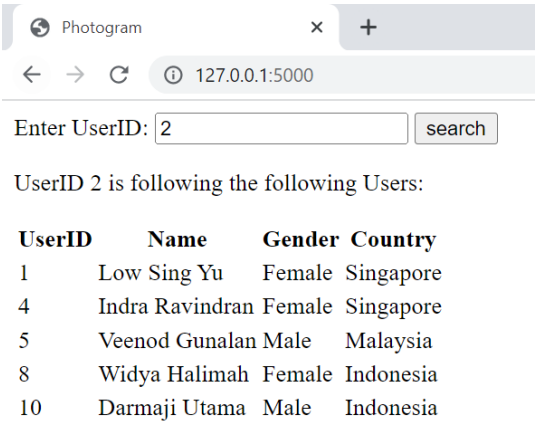
The data in `USERS.csv` is given in the following order:
`UserID, Name, Gender, Country`

The data in `FOLLOWING.csv` is given in the following order:
`UserID, Following_UserID (there may be multiple Following_UserID)`
The first data for each row in `FOLLOWING.csv` is the User ID of the user followed by 1 to many User IDs of people the user is following.

Write a Python program to read in the data from each file and store the data in the correct place in the database. [7]

Task 4.3
Write a Python program and the necessary files to create a web application that displays all the details of the people a particular user is following. The program should return a HTML document that enables the web browser to display a table with the required data. There should be a textbox to key in the User ID. No data should be shown if no User ID is given.





UserID	Name	Gender	Country
1	Low Sing Yu	Female	Singapore
4	Indra Ravindran	Female	Singapore
5	Veenod Gunalan	Male	Malaysia
8	Widya Halimah	Female	Indonesia
10	Darmaji Utama	Male	Indonesia

Save your Python program as:

`Task_4_3_<your name>_<centre number>_<index number>.py`

with any additional files/subfolders in a folder named:

`Task_4_3_<your name>_<centre number>_<index number>`

Run the web application.

Save the webpage output as:

`Task_4_3_<your name>_<centre number>_<index number>.html` [12]

Task 4.4
For Task 4.4, give your answer in the following Jupyter Notebook:

`TASK4_<your name>_<centre number>_<index number>.ipynb`

```
In [ ]: #Task 4.4
        Program code

        Output:
```

You heard about the many advantages of NoSQL database and decided to try migrating your data into a NoSQL database.

Write a Python program to extract all records of users and the people they are following from `photogram.db` or otherwise and insert into a PyMongo (NoSQL) database `photogram` under the collection `users`. Below is an example of a successfully inserted record in the `user` collection:

```
{'_id': <auto generated by PyMongo>, 'UserID': 1, 'Name': 'Low Si
ng Yu', 'Gender': 'Female', 'Country': 'Singapore', 'Following':
[2, 3, 5, 7]}
```
[4]

Save your Jupyter Notebook for Task 4.

End of Paper