MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
CAMBRIDGE ASSESSMENT INTERNATIONAL EDUCATION
General Certificate of Education Advanced Level
Higher 2

COMPUTING

9569/02

Paper 2 (Lab-based)

October/November 2022

3 hours

Additional Materials: Electronic version of `BOOK.txt` data file
Electronic version of `DATATOENCRYPT.txt` data file
Electronic version of `ENCRYPTEDMESSAGE.txt` data file
Electronic version of `LOAN.txt` file
Electronic version of `MEMBER.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 **6** marks out of 100 will be awarded for the use of common coding standards for programming style.

The numbers 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.

This document consists of **8** printed pages and **1** Insert.

Instruction to candidates:

Your program code and output for each of Task 1 to 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
```

The task is to implement a Caesar cypher encryption algorithm.

A Caesar cypher encodes each letter with a different letter. A 10-place Caesar cypher uses the ASCII value of each letter and adds the number 10 to it.

For example:

- The letter ‘A’ has the ASCII value 65. Adding 10 to the number will give the ASCII value 75. The character for 75 is ‘K’.
- When an uppercase letter’s code goes beyond ‘Z’ it returns to ‘A’. For example, the character ‘Z’ will be encrypted as ‘J’.
- When a lowercase letter’s code goes beyond ‘z’ it returns to ‘a’. For example, the character ‘x’ will be encrypted as ‘h’.
- Spaces ‘ ’ are replaced with the character ‘!’.

You will only need to convert letters and spaces. If the character is invalid, -1 is returned.

The following table shows the ASCII values of some of the characters.

Character	ASCII value
A	65
Z	90
a	97
z	122
‘ ’ (space)	32
!	33

In ASCII the letters follow on numerically. For example, the letter ‘A’ is 65, ‘B’ is 66, ‘C’ is 67 etc.

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 1.1
          Program code

          Output:
```

Task 1.1

This program will encrypt each letter by adding the number 10 to its ASCII value. Spaces will be replaced with '!'. If the character is invalid, -1 is returned.

Write program code for a function that takes a character as a parameter and returns the ASCII value of the new encrypted character. [5]

Task 1.2

Write program code to:

- read in a single character from the user
- call your function from **Task 1.1** with this character
- output its encrypted character or an appropriate message if the character is invalid. [3]

Test your program **four** times, with the following test data:

```
A
a
#
' '
```
(space) [2]

Task 1.3

The text file `DATATOENCRYPT.txt` contains a message that needs to be encrypted and then stored in a text file named `ENCRYPTEDMESSAGE.txt`

Write program code to:

- read the data from the text file `DATATOENCRYPT.txt`
- use your function from **Task 1.1** to encrypt each character
- store the encrypted message in the text file `ENCRYPTEDMESSAGE.txt`

Do **not** append invalid characters to the encrypted message. [7]

Test your program with `DATATOENCRYPT.txt`

Show the contents of `ENCRYPTEDMESSAGE.txt` after you have 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`

This task is to perform sorting algorithms on 100 integers held in a 1-dimensional list and then search for a value in the list.

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 2.1
          Program code

          Output:
```

Task 2.1

Write a function, `task2_1()` to:

- initialise a global 1-dimensional list
- generate 100 random integers between 1 and 100 (inclusive)
- store each integer in the list
- output the contents of the list. [2]

Test your program and show the output. [1]

Task 2.2

Write program code to:

- declare a procedure to implement a bubble sort
- implement a bubble sort to sort the unsorted 1-dimensional list from **Task 2.1** (you must **not** use a built-in function).

Write a program to:

- call your function from **Task 2.1**
- call the bubble sort procedure
- output the sorted list. [5]

Test your program and show the output. [1]

Task 2.3

Write program code to:

- declare a procedure to implement a merge sort
- implement a merge sort on the unsorted 1-dimensional list from **Task 2.1** (you must **not** use a built-in function).

Write a program to:

- ask the user to select which sorting algorithm they want to use
- loop until a valid choice is input
- call your function from **Task 2.1**
- call the appropriate sorting procedure
- output the sorted list. [7]

Test the program by first entering one invalid choice (for example, 'no') and then by entering one of the sorting methods (for example, 'bubble'). [1]

Task 2.4

Write program code to declare a recursive function to:

- take an integer value as one of its parameters
- implement a recursive binary search on the list from **Task 2.1**
- return the position of the integer in the list (if the integer appears more than once, then only one position needs to be returned)
- return -1 if the integer is **not** in the list.

Write a program to:

- read in an integer value from the user
- call the binary search function with the integer input
- output the returned position if the integer is in the list
- output 'Not found' if the integer is **not** in the list. [7]

Test the program with the smallest integer that is in the list and then with one integer that is **not** in the list. [2]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as:

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

A binary search tree is used to store 10 integer values between 0 and 999 (inclusive) in ascending numerical order.

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 data in the node.

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 store it in the correct position in the tree
- a recursive method to use in-order traversal to output the data in the tree
- a recursive method to use post-order traversal to output the data in the tree.

For the sub-task, 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

           Output:
```

Task 3.1

Write program code to declare the class `Tree` and its constructor. [4]

Write the recursive method to insert a new node into the tree. [6]

Write the main program to:

- declare a new instance of `Tree`
- generate 10 unique random integer values between 0 and 999 (inclusive)
- store each unique value as a new node in the tree using your method. [5]

Write program code to:

- declare the method to output the in-order traversal of the binary tree
- declare the method to output the post-order traversal of the binary tree.

Call the in-order and post-order methods using your tree structure. [7]

Test your program and show the output from each traversal. [2]

Save your Jupyter Notebook for Task 3.

4 Name your Jupyter Notebook as:

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

A library currently keeps paper records about its members, books and the books loaned. The library wants to create a suitable database to store the data and to allow them to run searches for specific data. The database will have three tables: a table to store data about the books, a table about the members and a table about the loans. The fields in each table are:

`Book`:

- `BookID` – unique book number, for example, 1234
- `Title` – the book title
- `Genre` – the type of book, for example, Drama, Sci-fi, Classic.

`Member`:

- `MemberNumber` – member's unique number, for example, 634
- `FamilyName` – member's family name
- `GivenName` – member's given name.

`Loan`:

- `LoanID` – the loan's unique number, for example, 12
- `MemberNumber` – the member's unique number
- `BookID` – the unique book number
- `DateLoaned` – the date that the book was taken out by the member
- `Returned` – `TRUE` if the book has been returned, or `FALSE` if it has **not** been returned.

For each of the sub-tasks 4.1 to 4.3, 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 4.1
          Program code

          Output:
```

Task 4.1

Write a Python program that uses SQL code to create the database `LIBRARY` with the three tables given. Define the primary and foreign keys for each table. [6]

Task 4.2

The text files `BOOK.txt`, `MEMBER.txt` and `LOAN.txt` store the comma-separated values for each of the tables in the database.

Write a Python program to read in the data from each file and then store each item of data in the correct place in the database. [5]

Task 4.3

Write a Python program to input a member's number and return the names of all the books that they have had out on loan, and whether each book has been returned. [5]

Test your program by running the application with the member number 200 [2]

Save your Jupyter Notebook.

Task 4.4

Write a Python program and the necessary files to create a web application, that displays the following data, about books that have **not** yet been returned:

- member's family name
- member's given name
- book title.

The program should return an HTML document that enables the web browser to display a table with the required data.

Save your Python program as:

`TASK_4_4_<your name>_<centre number>_<index number>.py`

with any additional files / subfolders in a folder named:

`TASK_4_4_<your name>_<centre number>_<index number>` [6]

Run the web application.

Save the webpage output as:

`TASK_4_4_<your name>_<centre number>_<index number>.html` [2]