MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
UNIVERSITY OF CAMBRIDGE LOCAL EXAMINATIONS SYNDICATE
General Certificate of Education Advanced Level
Higher 2

COMPUTING

9569/02

Paper 2 (Lab-based)

For Examination from 2020

SPECIMEN PAPER

3 hours

Additional Materials: Electronic version of `TESTDATA.txt` data file
Electronic version of `MONITORS.txt` data file
Electronic version of `PRINTERS.txt` data file
Electronic version of `LAPTOPS.txt` data file
Electronic version of `TASK3stack.txt` data file
Electronic version of `TASK3queue.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 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.

Singapore Examinations and Assessment Board

CAMBRIDGE International Examinations

Instruction to candidates:

Your program code and output for each of Task 1 to 3 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>_<centre number>_<index number>.ipynb`

1 The task is to read a single character from the keyboard, check that it is alphabetic and display the character in different number systems.

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:

In [2]: #Task 1.2
        Program code

        Output:

In [3]: #Task 1.3
        Program code

        Output:
```

Task 1.1

Write program code to:

- prompt the user to enter a single letter
- read a character from the keyboard
- test that it is a single letter in the ranges A–Z or a–z.
- display the character if it is in the correct range
- or display a suitable error message and allow the user to input the character again. [4]

Test your program with the following test data:

```
A
b
=
```
[2]

Task 1.2

Write program code to:

- input a letter
- input a number base and check that it is greater than 10 and less than 15
- convert the ASCII value of a single letter in denary
- convert the ASCII value to the number base input
- display the letter and both its values. [5]

Your screen display should look like this:

```
Letter            B
Denary            66
Number Base 11    60
```

Test your program with the following test data:

`B` and number base `11`
`m` and number base `14` [2]

Task 1.3

Extend your program to display a menu showing the conversions available, and allow the user to choose the conversion. Your menu should look like this:

```
1. Enter a letter
2. Convert to Denary
3. Convert to Base 11
4. Convert to Base 12
5. Convert to Base 13
6. Convert to Base 14
7. End
```
[4]

Test your program with the following test data:

```
X
Convert to Base 13
Convert to Base 11
y
Convert to denary
End
```

Download your program code and output for Task 1 as
`TASK1_<your name>_<centre number>_<index number>.ipynb` [3]

[Turn over

2 The task is to simulate part of the role of a file server by using a file that contains information about the files stored in a computer directory. The user can display the information sorted by file name or date modified, search for a file with a given name or search for any files stored on a given date. The file contains the following information for files stored in that directory. Each file name is different.

```
File_name
Date_modified
Size_KB
```

The file provided is `TESTDATA.txt`

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:

In [2]: #Task 2.2
        Program code

        Output:
```

Task 2.1

Write program code to:

- sort the contents of the files `TESTDATA.txt` into ascending order of file name using a quicksort algorithm
- display the sorted list and store the data in a new file,
  `TESTDATA_Filename_order_<your name>_<centre number>_<index number>.txt`

[8]

Save the file
`TESTDATA_Filename_order_<your name>_<centre number>_<index number>.txt` [2]

Task 2.2

Write program code to:

- sort and display the list in chronological date modified order
- search for a file with a given name, display the file name and size if found
- search for files modified on a given date, display the file name and date if found

All options must be written as separate routines. Any inputs must have meaningful prompts.
If file names or dates are not found, clear error messages must be displayed. File names searched for are exact matches.

Use a different sort algorithm to that used in **Task 2.1**. Identify the sort algorithm that has been used in the program comments. [13]

Provide evidence of testing that includes:

- sort in date order
- search for a file name that is contained in the file
- search for a file name that is not contained in the file
- display files modified on a given date
- display message showing that no files were modified on a given date

Download your program code and output for Task 2 as
`TASK2_<your name>_<centre number>_<index number>.ipynb` [5]

3 A programmer is writing a program to manipulate different data structures using Object-Oriented Programming (OOP).

The superclass, `DataStructure`, will store the following data:

- a linked list of the data items
- head pointer, pointing to the first element in the linked list
- tail pointer, pointing to the last element in the linked list

This class has one method to display all the current contents in the structure, in the order they are stored in the linked list.

The superclass is used to implement a stack and a linear queue.

The subclass `Stack` has the following methods:

- `insert(value)` appends the parameter to the stack.
- `delete()` returns and removes the next value in the stack.
- `display()` method should display the stack in reverse order (e.g. the most recently added element first) and should override the `DataStructure` display method.

The subclass `Queue` has the following methods:

- `insert(value)` appends the parameter value to the queue.
- `delete()` returns and removes the next value in the queue.
- `display()` method should display the queue contents in order (e.g. the earliest added element first) and should override the `DataStructure` display method.

Each method updates its appropriate pointers, and produces suitable errors (or returns different values) to indicate if the actions are not possible, e.g. if the structure is empty.

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
```

```
In [4]: #Task 3.4
        Program code
```

```
Output:
```

Task 3.1

Write program code for the superclass `DataStructure`. [3]

Task 3.2

Write program code for the subclass `Stack`.

Use appropriate inheritance and polymorphism in your designs. [5]

Task 3.3

Write program code for the subclass `Queue`.

Use appropriate inheritance and polymorphism in your designs. [5]

Task 3.4

The files `TASK3stack.txt` and `TASK3queue.txt` store data to test your program.

Write program code to:

- create a new stack and add the data in the file `TASK3stack.txt` to the stack
- create a new queue and add the data in the file `TASK3queue.txt` to the queue
- output the current contents of both the stack and queue
- remove and output two items from both the stack and queue
- output the contents of both the stack and queue after the removal of the items.

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

You are required to present the output of stack and queue both before and after the removal of items.

Download your program code and output for Task 3 as
`TASK3_<your name>_<centre number>_<index number>.ipynb` [9]

4 A large company currently keeps records on paper of all the computing equipment it owns. Every computer device has its information recorded when it is purchased.

The company decided to trial a database to manage its computing equipment records. It is expected that the database should be normalised.

When a computer device is purchased, the following information is recorded:

- `SerialNumber` – unique serial number of device
- `Type` – type of device (`'Monitor'`, `'Laptop'` or `'Printer'`)
- `Model` – model of device
- `Location` – where the device is used
- `DateOfPurchase` – date of purchase
- `WrittenOff` – whether the device is still in use (`'TRUE'` means device is written off and NOT in use, `'FALSE'` means device is still in use)

For monitors, the following extra information is recorded:

- `DateCleaned` – the last date the monitor was cleaned

For laptops, the following extra information is recorded:

- `WeightKg` – the weight in kilograms

For printers, the following extra information is recorded:

- `Toner` – type of toner required
- `DateChanged` – the last date the toner cartridge was changed

The information is to be stored in four different tables:

```
Device
Monitor
Laptop
Printer
```

Task 4.1

Create an SQL file called `TASK4_1_<your name>_<centre number>_<index number>.sql` to show the SQL code to create the database `equipment.db` with the four tables. The table, `Device`, must use `SerialNumber` as its **primary key**. The other tables must refer to the `SerialNumber` as a **foreign key**.

Save your SQL code as
`TASK4_1_<your name>_<centre number>_<index number>.sql` [5]

Task 4.2

The files `MONITORS.txt`, `LAPTOPS.txt` and `PRINTERS.txt` contain information about the company's monitors, laptops and printers respectively for insertion into the equipment database. Each row in the three files is a comma-separated list of information about a single device.

For `MONITORS.txt`, information about each monitor is given in the following order:
`SerialNumber,Model,Location,DateOfPurchase,WrittenOff,DateCleaned`

For `LAPTOPS.txt`, information about each laptop is given in the following order:
`SerialNumber,Model,Location,DateOfPurchase,WrittenOff,WeightKg`

For `PRINTERS.txt`, information about each printer is given in the following order:
`SerialNumber,Model,Location,DateOfPurchase,WrittenOff,Toner,DateChanged`

Write a Python program to insert all information from the three files into the `equipment` database, `equipment.db`. Run the program.

Save your program code as
`TASK4_2_<your name>_<centre number>_<index number>.py` [5]

Task 4.3

Write SQL code to show the serial number, model and the location of each monitor, with the date it was last cleaned. Run this query.

Save this code as
`TASK4_3_<your name>_<centre number>_<index number>.sql` [4]

Task 4.4

The company wants to filter the devices by `Location` and display the results in a web browser.

Write a Python program and the necessary files to create a web application that:

- receives a `Location` string from a HTML form, then
- creates and returns a HTML document that enables the web browser to display a table tabulating the `SerialNumber` and `Type` of devices still in use at that exact `Location`

Save your Python program as
`TASK4_4_<your name>_<centre number>_<index number>.py`
with any additional files / sub-folders as needed in a folder named
`TASK4_4_<your name>_<centre number>_<index number>`

Run the web application. Save the output of the program when "Office 51" is entered as the `Location` as `TASK4_4_<your name>_<centre number>_<index number>.html` [10]