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 2023

3 hours

Additional Materials: Electronic version of `imagefile.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.

This document consists of **11** printed pages, **1** blank page and **1** Insert.

Singapore Examinations and Assessment Board

Cambridge Assessment International Education

Instruction to candidates:

Your program code and output for each of Task 1 to 4 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`

A teacher currently keeps paper records about their students and the mark they have achieved in each test. The teacher wants to create a suitable database to store the data.

The database `TESTDATABASE` will have the following tables:

`Student`:

- `StudentID` – unique integer number per student, for example 12
- `GivenName` – the given name of the student
- `FamilyName` – the family name of the student
- `ClassName` – the class name of the student, for example CS1

`Test`:

- `TestID` – unique code for the test, for example EOU1
- `MaxMarks` – the maximum number of marks for the test

`Class`:

- `ClassName` – unique code for the class, for example CS101
- `YearGroup` – the year group of the class, for example 12

`Student_Test`:

- `StudentTestID` – unique integer, for example 123
- `StudentID` – the student's unique number
- `TestID` – the unique test code
- `Mark` – the mark the student achieved on the test
- `DateTest` – the date the test was taken, for example 05-05-2023

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

Write a Python program that uses SQL code to create the database `TESTDATABASE` with the four tables given. Define the primary and foreign keys for each table. [7]

Run your program to test the database has been set up correctly. [1]

Task 1.2

Write a Python program to insert the following information into the database `TESTDATABASE`

Student Mary Lim has the ID number 102.

Mary is in year 12 and class CS3.

Test PG1 has a maximum of 100 marks.

Mary took the test on 1 February 2023 and got 85 marks. This record should be stored using `StudentTestID` 123. [4]

Run your program to test the data has been entered correctly. [1]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as:

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

A program manages a circular queue of up to 4 items.

The class `QueueItem` contains two attributes:

- `ID` the string ID of the item
- `quantity` the integer number of items.

The class `QueueItem` contains the following methods:

- `getID()` that returns the ID of that object
- `getQuantity()` that returns the quantity of that object.

The class `Queue` contains four attributes:

- `queueList` a 1-dimensional list of `QueueItem` objects
- `headPointer` the index of the first item in the queue, initialised to 0
- `tailPointer` the index of the next free space in the queue, initialised to 0
- `numberItems` the quantity of items in the queue.

The class `Queue` has the following methods:

- a constructor that initialises `headPointer`, `tailPointer` and `numberItems` to appropriate values for an empty queue. It also initialises 4 null objects to `queueList`
- `enqueue()` that takes a `QueueItem` as a parameter and inserts it into the queue. It returns `True` if this is successful, or `False` otherwise
- `dequeue()` that returns the first item in the queue and returns the removed item, or `False` if empty
- `printQueue()` that outputs all current contents in the queue, from the first item in the queue to the last, without removing any of the items.

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 program code to declare the class `QueueItem` and its methods. [4]

Task 2.2

Write program code to declare the class `Queue` and its constructor. [3]

Task 2.3

Amend your code for Task 2.2 to declare the `Queue` method `enqueue()`

In a new cell, write program code to:

- initialise a new `Queue`
- take **five** sets of ID and quantity as input
- instantiate each set of inputs as a `QueueItem`
- call `enqueue()` with each new `QueueItem`
- output a suitable message if each addition was successful, or **not**. [7]

Test your program and show the output. [1]

Task 2.4

Amend your code for Task 2.2 to declare the `Queue` method `dequeue()`

In a new cell, write program code to:

- use `dequeue()` to remove **five** items
- output either the ID and quantity or an appropriate message if the queue is empty, for each item removed. [6]

Test your program and show the output. [1]

Task 2.5

Amend your code for Task 2.2 to declare the `Queue` method `printQueue()`

In a new cell, write program code to:

- enqueue **four** more objects
- call the method `printQueue()` [4]

Test your program and show the output. [1]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as:

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

A puzzle consists of a grid of 9 boxes by 9 boxes.

Each box has a number placed in it from 1 to 9 inclusive.

The puzzle is complete if each number is **not** repeated in each horizontal row, each vertical column, and each mini-square of 9 boxes (3 × 3).

Example 1. This puzzle is complete because each row only contains each number once. Each column only contains each number once. Each mini-square only contains each number once:

7	8	2	9	1	3	4	5	6
1	4	5	8	7	6	9	2	3
6	9	3	4	5	2	7	1	8
2	7	9	5	3	1	8	6	4
5	6	4	2	9	8	1	3	7
3	1	8	7	6	4	5	9	2
4	5	1	6	2	7	3	8	9
9	2	7	3	8	5	6	4	1
8	3	6	1	4	9	2	7	5

Example 2. This puzzle is incomplete because row 3 has the number 6 twice. This also means that column 4 has the number 6 twice, and the second mini-square on the top row has the number 6 twice.

7	8	2	9	1	3	4	5	6
1	4	5	8	7	6	9	2	3
6	9	3	6	5	2	7	1	8
2	7	9	5	3	1	8	6	4
5	6	4	2	9	8	1	3	7
3	1	8	7	6	4	5	9	2
4	5	1	6	2	7	3	8	9
9	2	7	3	8	5	6	4	1
8	3	6	1	4	9	2	7	5

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
```

```
Output:
```

Task 3.1

Write a function `inputPuzzle()` to allow the user to input a number into each of the boxes by:

- taking 9 rows of 9 digits as input from the user with a space between each digit, for example for one row: 1 2 3 4 5 6 7 8 9
- continually requesting a new row until the row is valid. A row is valid if it contains 9 digits that are all between 1 and 9 (inclusive). A row can have two or more of the same digits
- splitting each row into a list of digits
- appending each list of digits to a 2-dimensional list
- returning the 2-dimensional list.

[7]

Test your program by entering the data for the puzzles shown in Example 1 and Example 2. Store the puzzles in variables named `example1` and `example2` respectively. [1]

Task 3.2

Write program code to declare a function that checks each row of the puzzle to make sure it is complete horizontally (contains each of the numbers 1 to 9 only once). The function should return −1 if the puzzle is complete horizontally, or the row number where there is a duplicate. [3]

Test your program with both of the puzzles you stored in Task 3.1 by calling the function with each puzzle and outputting an appropriate message depending on the return value. [3]

Task 3.3

Write program code to declare a function that checks each column of the puzzle to make sure it is complete vertically (contains each of the numbers 1 to 9 only once). The function should return −1 if the puzzle is complete vertically, or the column number where there is a duplicate. [3]

Test your program with both of the puzzles you stored in Task 3.1 by calling the function with each puzzle and outputting an appropriate message depending on the return value. [3]

Task 3.4

Write program code to declare a function that checks each of the 9 mini-squares to make sure they are all complete (contain each of the numbers 1 to 9 only once). The function should return `True` if all mini-squares are complete, or `False` if any mini-squares are **not** complete. [4]

Test your program with both of the puzzles you stored in Task 3.1 by calling the function with each puzzle and outputting an appropriate message depending on the return value. [3]

Save your Jupyter Notebook for Task 3.

4 Name your Jupyter Notebook as:

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

Run-length encoding is a compression method that can be used to compress image files. It counts the number of consecutive pixels of each colour, then stores a code for the colour and the number of times it occurs. For example, instead of storing the code for black three times, it will store the code once, then the number 3. In some images each pixel will be a different colour. In some images all of the pixels might be the same colour. For this task, there will be no more than 9 consecutive pixels of the same colour.

For example:

Image:

black	green	yellow
yellow	yellow	yellow
black	black	black

Colour codes:

Colour	Code
black	01
green	10
yellow	11

This image has 9 pixels (3 × 3). Each colour in this image has a bit depth of 2, this means each colour is represented by 2 bits.

Before compression, the image will be stored as:

011011111111010101

When compressed the image will be stored as:

011101114013

For each of the sub-tasks 4.1 and 4.2, 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

The text file `imagefile.txt` stores the colour of each pixel for an image:

- The image is 9 pixels wide by 9 pixels high
- Each colour is stored with a bit depth of 3. This means each colour is stored in 3 bits.
- Each pixel is written on a new line in the text file. For example the first line has the value 000. 000 is the colour code for the first pixel in the image.

Write program code to:

- declare a function that takes a file name as a parameter and reads in the data from the file
- declare a function that takes a list of colours as a parameter and compresses the data
- declare a function that takes the compressed data as a parameter and writes this to a text file with the name `compressedimage.txt`
- call the functions from the program with input `imagefile.txt`

The compressed data should be stored as one string of digits. [7]

Run your program. [1]

Task 4.2

The program also needs to decompress a compressed image file.

Write program code to:

- declare a function that takes a file name as a parameter and reads in the compressed image data from the file
- declare a function that takes the data for a compressed image as a parameter and decompresses the data
- declare a function that takes the data for a decompressed image as a parameter and writes this data to a text file with the name `decompressedimage.txt`
- declare a menu that allows the user to enter a file name and choose between compressing an image or decompressing an image. The appropriate function or functions are called. [8]

Run your program. [1]

Save your Jupyter Notebook for Task 4.

Task 4.3

Write a Python program and the necessary files to create a web application that displays the colour of each pixel in the text file `decompressedimage.txt`

The table shows the colour of each binary code:

Colour	Code
red	000
white	001
yellow	010
blue	011
black	100
green	110

The image can be displayed using a table that has 9 rows and 9 columns. Each pixel is defined by **either** changing the background colour of that cell **or** writing the name of the colour in the cell.

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

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>` [8]

Run the web application.

Save the web page output as:

`TASK_4_3_<your name>_<centre number>_<index number>.html` [2]

BLANK PAGE