NANYANG JUNIOR COLLEGE

JC2 PRELIMINARY EXAMINATIONS

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **18th August 2021**

3 Hours

Additional Materials: Removable storage device
Electronic version of `TIDES.txt` data file
Electronic version of `bookstore.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 **10** printed pages.

© NYJC 2021

[Turn over

Instruction to candidates:

Your program code and output for each of Task 1 to 3 should be saved in a single .ipynb file. For example, your program code and output for Task 1 should be saved as

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

1 Name your Jupyter Notebook as

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

A text file, `TIDES.TXT`, contains the low and high tide information for a coastal location for each day of a month. Each line contains tab-delimited data that shows the date, the time, whether the tide is high or low and the tide height in metres.

Each line is in the format:

`YYYY-MM-DD\tHH:mm\tTIDE\tHEIGHT\n`

- The date is in the form YYYY-MM-DD, for example, 2019-08-03 is 3rd August, 2019
- The time is in the form HH:mm, for example, 13:47
- TIDE is either HIGH or LOW
- HEIGHT is a positive number shown to one decimal place
- `\t` represents the tab character
- `\n` represents the newline character

The text file is stored in ascending order of date and time.

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 program code to:

- read the tide data from a text file
- find the highest high tide and print this value
- find the lowest low tide and print this value

Use `TIDES.TXT` to test your program code. [7]

Save your Jupyter Notebook for Task 1.

Task 1.2

The tidal range is the difference between the heights of successive tides; from a high tide to the following low tide or from a low tide to the following high tide.

Amend your program code to:

- output the largest tidal range and the date on which the second tide occurs
- output the smallest tidal range and the date on which the second tide occurs

Use `TIDES.TXT` to test your program code. [4]

(Total: 11)

2 Name your Jupyter Notebook as

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

The task is to implement a todo list using a linkedlist data structure.

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

The class `TodoList` represents a LinkedList and has the following attributes:

- `__head` – a pointer to the first node of the LinkedList; if empty, it has a value of `None`
- `__tail` – a pointer to the last node of the LinkedList; if empty, it has a value of `None`

`TodoList` has the following methods defined on it:

- `add(item)` – wraps `item` in a `TodoItem` instance, and adds it to the end of the LinkedList
- `remove(item)` – removes the first `TodoItem` containing `item` from the LinkedList
- `list()` – returns a Python list containing each `item` in the TodoList

The class `TodoItem` represents a Node of the LinkedList and has the following attributes:

- `title` – a short description of the todo item
- `__next` – a pointer to the next node in the LinkedList; if this is the last node, it has a value of `None`

`TodoItem` has the following methods defined on it:

- `link_to(todoitem)` – links this `TodoItem` instance to `todoitem`, another instance of the `TodoItem` class

Implement the above classes. [13]

Task 2.2

Add the following items to a new `TodoList`:

- “Buy milk”
- “Buy flour”
- “Buy eggs”
- “Bake cake”

Display the contents of the `TodoList`. [7]

Task 2.3

Remove the following items from the `TodoList`:

- “Buy milk”
- “Buy eggs”

Display the contents of the `TodoList`. [3]

Save your Jupyter Notebook for Task 2.

(Total: 23)

3 Name your Jupyter Notebook as

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

The task is to write a function that takes a sequence of characters representing a colour, and translates the colour into a different number base.

8-bit colours are represented with three numbers, indicating the level of the colours red (R), green (G), and blue (B) respectively. Each number is an integer from 0 to 255. 255 represents the fully saturated colour, while 0 represents zero saturation (black).

In HTML, these colours may be represented using hex code as well. In hex code, the R, G, and B values are converted to hexadecimal. Hex codes begin with the symbol '#' followed by the three R, G, and B hexadecimal values.

For example, the hex code #0A0B0C represents a colour with RGB values 10, 11, and 12 respectively.

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 called `task3_1(hex)` that:

- takes `hex`, a string representing a hex code, beginning with a '#' symbol followed by three valid hexadecimal values between 00 and FF
- returns and displays either:
  - a 3-integer tuple representing RGB values

  or
  - the error message, `"invalid data"` [5]

Test the function fully with suitable test data.

For example,

```
task3_1("#FFFFFF")
```

should return and display `(255, 255, 255)` [3]

Task 3.2

Some image programs do not represent colours using 8-bit integers. Instead, they represent them as a normalised float value. In this representation, a value of 1.0 represents the fully saturated colour and a value of 0 represents zero saturation (black).

Write a second function `task3_2(rgb)` that:

- takes a 3-integer tuple `rgb` representing RGB values
- returns and displays either:
  - a 3-float tuple representing normalised RGB

  or
  - the error message, `"invalid data"` [5]

Test the function fully with suitable test data.

For example,

```
task3_2((128, 128, 128))
```

should return and display `(0.50196, 0.50196, 0.50196)`

```
task3_2((255, 255, 255))
```

should return and display `(1.0, 1.0, 1.0)` [3]

Task 3.3

Image filters are functions that take in image data and change the RGB values of its colours according to an algorithm. The algorithm for converting an image to grayscale calculates the average of the RGB values and sets the R, G, and B values to this average.

Write a third function `task3_3(hex)` that:

- takes `hex`, a string representing a hex code
- returns and displays a 3-float tuple representing normalised RGB of the colour converted to grayscale [4]

Test the function fully with **two** suitable values.

For example,

```
task3_3("#FF8000")
```

should return and display `(0.5, 0.5, 0.5)` [2]

Save your Jupyter Notebook for Task 3.

(Total: 22)

4 A bookstore uses a text file to store data about its inventory of books. The bookshop carries two kinds of books: printed books and virtual books. The bookshop wishes to transfer this information into a database.

The bookshop also wishes to create an online bookstore that allows users to add books to a shopping cart for purchase.

Task 4.1

Create an SQL file called `TASK4_1_<centre number>_<index number>.sql` to show the SQL code to create database `bookstore.db` with three tables: `Book`, `Printed`, and `Virtual`. The Printed and Virtual tables represent physical and virtual books respectively, and stores properties unique to each type of book.

The `Book` table will have the following fields:

- `BookID` – the primary key, an integer value
- `Title` – the title of the book
- `Price` – the price of the book, in cents
- `Type` – the type of book: "physical" or "virtual"

The `Printed` table will have the following additional field:

- `Weight` – the weight of the book

The `Virtual` table will have the following additional field:

- `DownloadLink` – the download link for the book

Save your SQL code as

`TASK4_1_<your name>_<centre number>_<index number>.sql` [6]

Task 4.2

Python programming language and object-oriented programming will be used to implement the online bookstore and shopping cart on a web page.

The class Book will store the following data:

- `title` – stored as a string
- `price` – stored as an integer

The class `Cart` will store the following data:

- `items` – stored as a list of `Book` objects

The class `Cart` has a method defined on it:

- `total_price()` – returns an integer representing the total price of books in the cart

Save your program code as

`TASK4_2_<your name>_<centre number>_<index number>.py` [6]

The `PrintedBook` class inherits from `Book`, and stores the following additional data:

- `weight` – stored as an integer

The `VirtualBook` class inherits from `Book`, and stores the following additional data:

- `download_link` – stored as a string

Add your program code to

`TASK4_2_<your name>_<centre number>_<index number>.py` [3]

The text file, `bookstore.txt`, contains data items for books stocked by the bookstore. Each data item is separated by a comma, with each book's data on a new line as follows:

- book title
- price
- type
- weight
- download link

Write program code to read in the information from the text file, `bookstore.txt`, creating an instance of the appropriate class for each book (either `PrintedBook` or `VirtualBook`). [4]

Write program code to insert all information from the file into the `bookstore.db` database.

Run the program.

Add your program code to

`TASK4_2_<your name>_<centre number>_<index number>.py` [8]

Task 4.3

The data from the text file, `bookstore.txt`, is to be used to implement a shopping cart in a web browser.

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

- displays a list of books stocked by the bookstore
- enables the user to add books to a shopping cart using an ID
- displays the contents of the shopping cart
- shows the total price of items in the shopping cart

For each book displayed the web page should include the:

- book ID
- book title
- price

Save your program as

`TASK4_3_<your name>_<centre number>_<index number>.py`

with any additional files / sub-folders as needed in a folder named

`TASK4_3_<your name>_<centre number>_<index number>` [7]

Run the web application and add the following books to the shopping cart:

- Title: "Northanger Abbey", Price: 13.99, Type: Physical, Weight: 178g
- Title: "War and Peace", Price: 17.49, Type: Physical, Weight: 432g
- Title: "Computer Programs", Price: 20.99, Type: Virtual, Link: https://mybookstore.com/dJHtFy
- Title: "Data Science", Price: 14.99, Type: Virtual, Link: https://mybookstore.com/fJynJk

Save the output of the program as

`TASK4_3_<your name>_<centre number>_<index number>.html` [4]

(Total: 38)