Name:_____ **Class:**_____

JURONG PIONEER JUNIOR COLLEGE
JC 1 Year – End Examination 2024

COMPUTING **9569/02**

Higher 2 **23 September 2024**

Paper 2 (Practical) **2 hours**

Additional materials: Cover Page
Electronic version of `TASK1.txt` data file
Electronic version of `TASK3.txt` data file
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** the questions.

All tasks must be completed 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 from the examination venue.

Approved calculators are allowed.

You are advised to 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 **3** marks out of 60 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 task.

The total number of marks for this paper is **60**.

This document consists of **6** printed pages and **2** blank pages

BLANK PAGE

Instructions 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 class>_<your name>.ipynb`.

1 JP Bookstore maintains a list of books in a text file called `TASK1.txt`. Each line in the file contains information about a single book in the following format:

```
<Title>,<Genre>,<Price>
```

Task 1.1

Write a function `read_books_data()` to:

- read the contents of `TASK1.txt`
- return a list of lists.

Each inner list should represent a book with elements: title, genre, and price with their appropriate data types. [4]

Test your program and show the returned list of lists. [1]

Task 1.2

Write a function `get_books_by_genre(books, genre)` to:

- take the list of books, as returned by the function from **Task 1.1**, and a genre STRING as input arguments
- return a list of book titles that belong to the specified genre. [5]

Test your program with **`"Fiction"`** and show the returned list. [1]

Task 1.3
Write a function `get_expensive_books(books, price_threshold)` to:

- take the list of books, as returned by the function from **Task 1.1**, and a price threshold DECIMAL as input arguments
- return a list of book titles that have a price greater than or equals to the specified price threshold. [1]

Test your program with **`15.99`** and show the returned list. [1]

Task 1.4

Write a program code to:

- display a menu with the following options:
  `1) Display Books by Genre`
  `2) Display Books by Price`
  `3) Quit`
- ask the user to select option `1`, `2` or `3`
- loop until a valid choice is input
- execute your function from **Task 1.1**
- execute the appropriate function from **Task 1.2** or **Task 1.3**
- output the titles. [6]

The following is a sample run of the program:

```
1) Display Books by Genre
2) Display Books by Price
3) Quit

Enter your option (1, 2 or 3): 1
Enter the genre: Fiction

To Kill a Mockingbird
The Catcher in the Rye

1) Display Books by Genre
2) Display Books by Price
3) Quit

Enter your option (1, 2 or 3): 2
Enter the price: 15.99

To Kill a Mockingbird
1984
Brave New World

1) Display Books by Genre
2) Display Books by Price
3) Quit

Enter your option (1, 2 or 3): 3
```

Test the program by first entering option `1` (genre = `"Fiction"`), and then by entering option `2` (price = `15.99`) and lastly by entering option `3`. [1]

Save your Jupyter Notebook for Task 1.

2 A financial technology company needs a system that can handle binary and hexadecimal representations of financial data. The system should be able to convert between different numerical bases for efficient data processing and storage.

Task 2.1

Write a function `binary_to_denary(binary_str)` to:

- take a binary STRING as its parameter
- validate the binary STRING
- return -1 if it is invalid
- otherwise, return its corresponding denary INTEGER. [5]

Test your program with the following function calls and show the output:

- `binary_to_denary("1100100")`
- `binary_to_denary("11001AB")` [2]

Task 2.2

Write a function `convert_to_base(denary_int, base)` to:

- take a denary INTEGER and a base INTEGER (either 2 for binary or 16 for hexadecimal) as its parameters
- return the corresponding STRING representation in that base. [8]

Test your program with the following function calls and show the output:

- `convert_to_base(100, 2)`
- `convert_to_base(168, 16)` [2]

Save your Jupyter Notebook for Task 2.

3 You are developing a simple system to manage a playlist for a music streaming app. Each song in the playlist needs to be represented as a node in a linked list. Implement a dynamic linked list where you can add, remove, and display songs in the playlist.

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

The class `Node` contains three properties:

- `title` is the title of the song
- `artiste` is the artiste of the song
- `next` points to the next `Node` object in the linked list.

The class `Node` contains the following method:

- a constructor to assign `title` and `artiste` to `""` and `next` to `None`.

The class `LinkedList` contains the following property:

- `head` points to the first Node object in the linked list.

The class `LinkedList` contains the following methods:

- a constructor to assign `head` to `None`
- an `addNode` method to take the `Node` object parameter and add it to the tail of the linked list
- a `removeNode` method to take the song title as its parameter and remove the node with that title
- a `display` method to traverse the linked list from its head to tail and output the `title` and `artiste` in neat columns.

You may assume the titles are unique.

Task 3.1

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

Task 3.2

Write program code to declare `LinkedList` class and its methods. [14]

Task 3.3

Use the provided text file `TASK3.txt` to instantiate a linked list and four nodes. Copy the code from the text file to your main program.

Write the code to invoke the following methods in the main program:

- `addNode(node1)`
- `addNode(node2)`
- `addNode(node3)`
- `removeNode(“Just the Way You Are”)`
- `addNode(node4)`
- `removeNode(“24K Magic”)`
- `display()` [2]

Test your program and show the output. [1]

Save your Jupyter Notebook for Task 3.

END OF PAPER

BLANK PAGE