Name:_____ **Class:**_____

JURONG PIONEER JUNIOR COLLEGE

JC2 Mid-Year Examination 2023

COMPUTING **9569/02**
Higher 2 **26 June 2023**

Paper 2 (Lab-based) **2 hours**

Additional materials: Electronic version of `Task2_datetime.py` file
Electronic version of `TASK1.txt` data file
Electronic version of `TASK2.txt` data file
Electronic version of `gallery.db` file
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** the 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.

The number of marks is given in brackets [ ] at the end of each task.

The total number of marks for this paper is **65**.

This document consists of **7** printed pages.

[Turn over

Instructions to candidates:

Your program code and output for each of Task 1 and 2 should be saved in a single `.ipynb` file. For example, your program code and output for Task 1 should be saved as:

`TASK1_<your class>_<your name>.ipynb`.

Code each sub-task in one cell and add a comment statement, at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to.

1 Name your Jupyter Notebook as:
`TASK1_<your class>_<your name>.ipynb`

A program is to be written to implement an inventory of products using object-oriented programming (OOP). The inventory shows the products available for purchase.

Each product is given a `stockID`, `name`, and `price`.

The base class will be called `Product` and is designed as follows:

Product

`stockID: INTEGER`
`name: STRING`
`price: REAL`

`Constructor (stockID: INTEGER, name: STRING, price: REAL)`
`set_stockID(s: INTEGER)`
`set_name(n: STRING)`
`set_price(p: REAL)`
`get_stockID(): INTEGER`
`get_name(): STRING`
`get_price(): REAL`

Task 1.1
Write program code to define the class `Product`. **[6]**

The inventory can have products with extra information if they are perishable items. The `PerishableProduct` class inherits from the `Product` class, extending it to have an `expiry_date`, designed as follows:

PerishableProduct: Product
`expiry_date: STRING`

`Constructor (date: STRING, stockID: INTEGER, name: STRING, price: REAL)`
`set_date(d: STRING)`
`get_date():STRING`

A perishable product that expires in three days or less will have a 20% discount off its price (rounded to two decimal places). The `PerishableProduct` class should extend the `get_price()` method to return the final price (i.e.: with or without discount).

Task 1.2

The `datetime` library is built into Python and can be used to get today's date, instantiate a new `date` object and calculate the difference (days) between two `date` objects. Example code is shown in `Task2_datetime.py`.

Write program code to define the class `PerishableProduct`. Implement inheritance and polymorphism. Use the `datetime` module to solve the task. **[6]**

A text file `TASK1.txt` contains information for the products available. Each line contains comma-delimited data that shows the type of product, stockID, name, price, and expiry date (if applicable).

Each line is in one of the following formats:

```
P,1111,Milk,3.50,2023-06-29
NP,2354,Saucepan,19.90
```

The date is in the form `YYYY-MM-DD`.

Task 1.3

In the main program, write the program to:

- create an empty list of objects `lst`, and
- add each of the record in the text file to the list (as either `Product` or `PerishableProduct` object).

[5]

Task 1.4

Write a program to implement `Display(lst)` procedure to:

- traverse the list of objects from Task 1.3,
- output the `stockID`, `name`, `price` (rounded to two decimal places) in three neat columns, and
- the price should reflect any discount applied.

[5]

Python does not print tail zeros by default. You may use the `f-strings` formatting to include tail zeros in the output.

[Turn over

Sample code:

```
test = round(5.90234,2)
print(test)
print(f"{test:.2f}")
```

Sample output:

```
5.9
5.90
```

Task 1.5

Test your program by calling `Display` procedure. **[2]**

Save your Jupyter notebook for Task 1.

2 A **Binary Search Tree** is used to **store hexadecimals in ascending order**. The node and tree are implemented using Object-Oriented Programming (OOP).

The class `Node` contains three properties:

- `left_pointer` points to the left subtree,
- `right_pointer` points to the right subtree, and
- `data` is the data in the node.

The class `Node` contains the following method:

- a constructor that sets the left pointer and right pointer to `None`, and the data to its parameter.

The class `Tree` contains one property:

- `root` points to the root node of the tree.

The class `Tree` contains the following methods:

- a constructor to initialise the root property to `None`,
- an `insert` method to take the parameter, create a new `Node` and store it in the correct position in the tree, and
- an `inorder_traversal` method to use in-order traversal to return a list containing the data in the tree.

You are given the program code in `TASK2.txt` for:

- the class `Node` and its constructor, and
- the class `Tree` and its constructor.

Task 2.1

A hexadecimal number system, often referred to as 'hex', uses base 16 and the digits 0 to 9 and the letters A to F. Write program code for `hexToDen` that will convert a hexadecimal string to denary (base 10) and return the integer result. **[5]**

Task 2.2

In Jupyter Notebook:

- copy and paste the codes from `TASK2.txt`, and
- write the `insert` method of the `Tree` class. **[6]**

Write the main program to:

- declare a new instance of `Tree`,
- declare the list, `hexlst = ['A1','22','77','B','88','F1']`, and
- insert each hexadecimal values into the Tree according to the order in the list. **[2]**

Task 2.3

Write program code to:

- declare the `inorder_traversal` method to return a list containing the in-order traversal of the binary tree, and
- call the in-order traversal method using your tree structure, and display the list returned by the traversal. **[7]**

Save your Jupyter Notebook for Task 2.

[Turn over

3 An art gallery uses a SQL database to store its inventory of artworks. Some artwork data is stored in a database `gallery.db`, provided with this question.

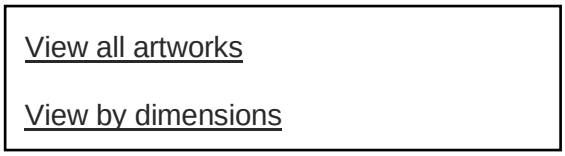
The database has `Artwork` table.

The `Artwork` table will have the following fields:

- `artworkID` – the artwork's unique number
- `Title` – the title of the artwork
- `Price` – the price, an integer value
- `Artist` – the name of the artist who created the piece
- `Breadth` – the width of the artwork in cm, an integer value
- `Length` – the length of the artwork in cm, an integer value
- `Weight` – the weight of the artwork in kg, a decimal value

Task 3.1

Write a Python program and the necessary files to create a web application. The web application has the following menu options, where each option is a hyperlink:



Save your Python program as:
`TASK3_1_<your class>_<your name>.py`

with any additional files / sub-folders as needed in the appropriate subfolder(s). **[5]**

Task 3.2

Write an SQL query that displays a list of all artworks stocked by the gallery on the home page. For each artwork displayed the web page should include the:

- `artworkID`
- `Title`
- `Artist`
- `Price`

Save your SQL code as:
`TASK3_2_<your class>_<your name>.sql`

[2]

Task 3.3
Write a Python program and the necessary files to display the data on a web page that is accessed from the "View all artworks" menu option from Task 3.1.

Save your Python program as:
`TASK3_3_<your class>_<your name>.py`

with any additional files / sub-folders as needed in the appropriate subfolder(s). **[5]**

Task 3.4

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

- display a web form,
- allows user to enter the maximum breadth and length, and
- displays the physical artworks with dimensions less than or equal to the values entered.

The web application should be accessed from the 'View by dimensions' option in the menu.

For each artwork displayed the web page should include the:

- `artworkID`
- `Title`
- `Artist`
- `Price`

Save your Python program as:
`TASK3_4_<your class>_<your name>.py`

with any additional files / sub-folders as needed in the appropriate subfolder(s).

Test your program with the follow values in cm:
`Breadth` – 100
`Length` – 100

Run the web application and save the output as
`TASK3_4_<your class>_<your name>.html`

[9]

END OF PAPER