

Name: _____

Class: _____



JURONG PIONEER JUNIOR COLLEGE

JC2 Preliminary Examination 2022

**COMPUTING
Higher 2**

9569/02

31 August 2022

Paper 2 (Practical)

3 hours

Additional materials:

Cover Page
Electronic version of RECORDS . csv data file
Electronic version of RENTAL . JSON data file
Electronic version of CUSTOMER . TXT data file
Electronic version of PRODUCT . TXT data file
Electronic version of CUSTOMERRENTAL . TXT data file
Electronic version of PRODUCTRENTAL . TXT data 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.

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 task.

The total number of marks for this paper is 100.

This document consists of **10** printed pages and **1** blank page.

[Turn over

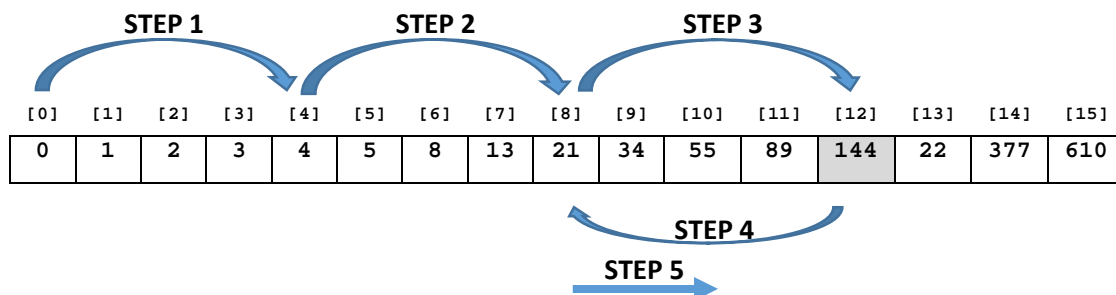
Instructions to candidates:

Your program code and output for each of Task 1 to 2 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 The Jump Search algorithm was designed as it is not always necessary to traverse and compare every item in the array like how Linear Search does it. Jump Search works on sorted arrays. The fundamental idea is to perform fewer checks by jumping ahead in fixed steps also known as a block, and not searching for the search key item by item, without the possibility of skipping.

For example, if Jump Search is applied to search for x in an array arr of size n , with a block (to be jumped) of size m . Search is performed at indices that are multiples of m (i.e.: $arr[0], arr[m], arr[2m] \dots arr[km]$ and so on). Once the interval ($arr[km] < x < arr[(k+1)m]$) is found, a linear search operation from the index km is performed.

Consider the following array: $[0, 1, 2, 3, 4, 5, 8, 13, 21, 34, 55, 89, 144, 22, 377, 610]$. The size of the array is 16, with block of size 4, and search value of 55 (i.e.: $x = 55$ and $m = 4$).



The Jump Search will find the value of 55 with the following steps:

STEP 1: Jump from index 0 to index 4.

STEP 2: Jump from index 4 to index 8.

STEP 3: Jump from index 8 to index 12.

STEP 4: Since the element at index 12 is greater than 55, jump back to index 8.

STEP 5: Perform a linear search from index 8 to search the element 55.

Task 1.1

Write program code for a function `JumpSearch(arr, x, m)` that implements an **iterative** Jump Search. The function returns the integer index where `x` is found in `arr` or `-1` if `x` is not found. You may assume there are no duplicate values stored in `arr`.

Test your function with the following values:

| Case | Identifier | Values |
|------|------------------|---|
| 1 | <code>arr</code> | [0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 999] |
| | <code>x</code> | 55 |
| | <code>m</code> | 4 |
| 2 | <code>arr</code> | [0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 999] |
| | <code>x</code> | 1000 |
| | <code>m</code> | 4 |

[10]

Task 1.2

Write program code for a function `JumpSearchRecursive(arr, x, m, currentLastIndex)` that implements a **recursive** Jump Search. The function returns the integer index where `x` is found in `arr` or `-1` if `x` is not found. You may assume there are no duplicate values stored in `arr`.

The linear search performed, after the interval is found, may be iterative.

Test your function with the two test cases from Task 1.1.

[6]

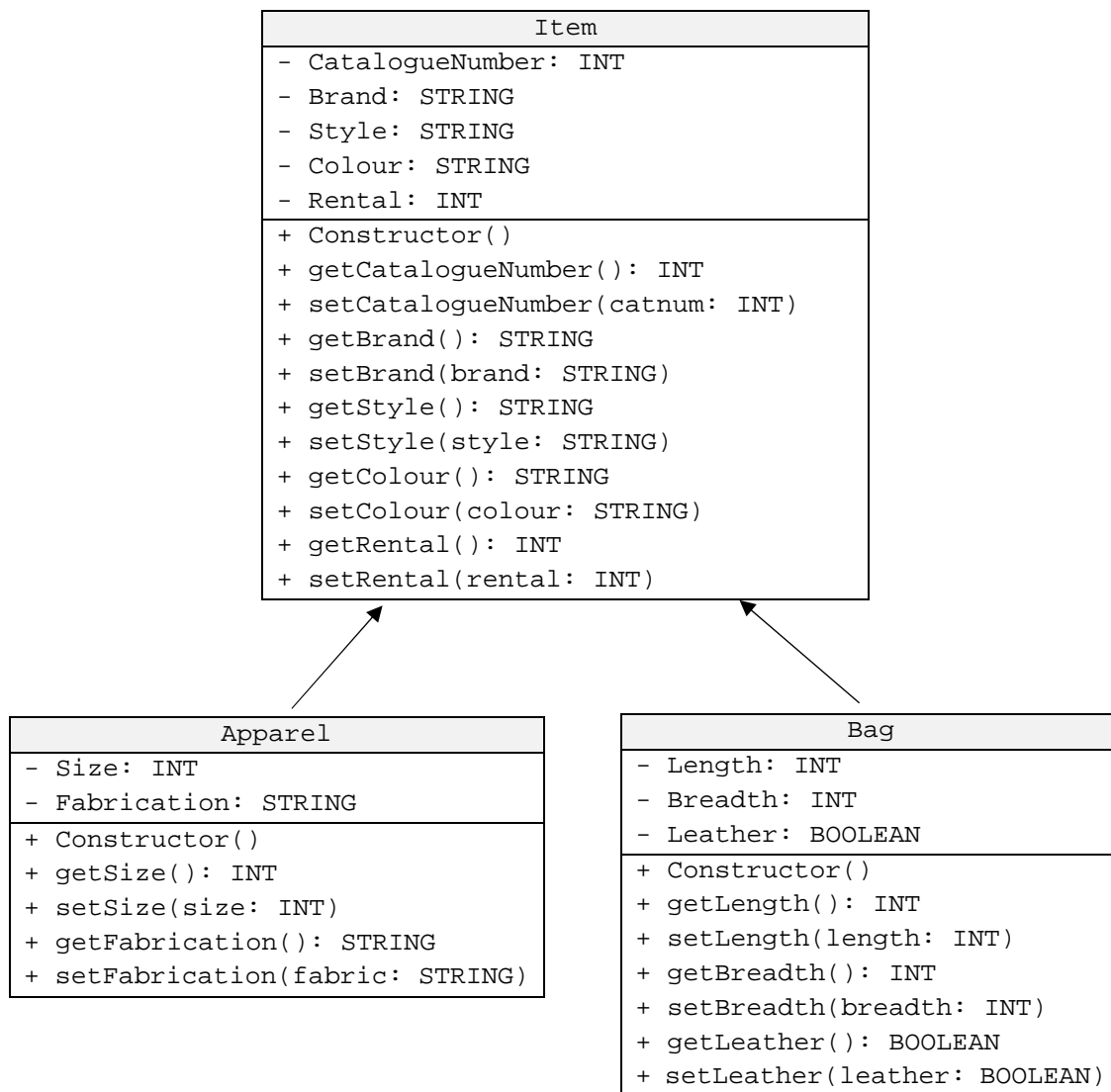
Download your program code for Task 1 as

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

- 2 Style Theory is a company where customers pay a fee to rent designer apparel and bags. A computer program running on the company's server uses a hash table to store details about who is currently renting an item. A hash table is a data structure that contains a collection of key-value pairs where the value is accessed via the associated key. The key used is the catalogue number of the item and the value is the email address of the customer.

Task 2.1

Write program code to define the classes `Item`, `Apparel` and `Bag`. `Item` is the parent class of `Apparel` and `Bag` classes.



[8]

Task 2.2

The size of the hash table array is 100. In addition, the hashing algorithm that has been used is $\text{CatalogueNumber} \bmod 100$. **Linear probing** is implemented to handle collisions.

A flat file `RECORDS.csv` stores the current rental records. Write the program code to:

- Initialise `hashtable[0:99]`.
- For each record in the flat file, instantiate an appropriate object (of either `Apparel` or `Bag ADT`).
- The rental information is stored in a tuple. The first element in the tuple is the object instantiated and the second element in the tuple is the customer's email address. For example: (`Apparel object`, `"mundepi@gmail.com"`).
- Insert the tuple into `hashtable`.

[8]

Task 2.3

Display in neat columns, the index, catalogue number and customer email stored from index 0 to 9 of `hashtable`.

[2]

Task 2.4

Hence, write program code for a function `HashTableSearch(hashtable, CatalogueNumber)` that returns the email address of the customer renting the item associated with `CatalogueNumber`, or "Not Found" if the item is not in `hashtable`.

[8]

Task 2.5

Test `HashTableSearch` function with the following `CatalogueNumber`:

- 1399
- 1220

[2]

Download your program code for Task 2 as

`TASK2_<your class>_<your name>.ipynb`

- 3 Style Theory decides to manage its rental using a NoSQL database instead. Information about the items is stored in the JSON file `RENTAL.JSON`.

The following fields are recorded:

- catalogue number,
- brand,
- category (i.e.: apparel or bag),
- daily rental fee,
- size (for apparel only),
- customer's email
- start date, and
- end date.

Task 3.1

Write program code to import the information from the JSON file into a MongoDB database. Save the information under the `Rental` collection in the `StyleTheory` database. Ensure that the collection only stores the information from the JSON file.

[2]

Save your program code as

`TASK3_<your class>_<your name>.py`

Task 3.2

Write program code for a user to insert information of a new rental or update the end date of an existing rental. Display the following user menu for the user:

```
Style Theory
Option 1 - Insert new rental
Option 2 - Update existing rental
Option 3 - Quit
Enter your option (1, 2, or 3):
```

If Option 1 is entered, get user input for all relevant fields and insert the new document to the collection. Size data input is applicable for apparel only.

Otherwise, if Option 2 is entered, get user input for the catalogue number, start date and new end date and update the end date of the existing document.

Run your program to insert the following documents:

| Field | Document 1 | Document 2 |
|------------------|---------------|---------------|
| Catalogue Number | 1400 | 1375 |
| Brand | Trioon | Gucci |
| Category | Apparel | Bag |
| Daily Rental Fee | 20 | 50 |
| Size | 8 | |
| Customer Email | liu@gmail.com | liu@gmail.com |
| Start Date | 2022-01-10 | 2022-01-10 |
| End Date | 2022-01-14 | 2022-01-14 |

[Turn over

Run your program to update the following document:

| Field | Document 1 | Document 2 |
|------------------|------------|------------|
| Catalogue Number | 1371 | 1266 |
| Start Date | 2022-01-03 | 2022-01-03 |
| End Date | 2022-01-05 | 2022-01-05 |

[10]

Add your program code to

TASK3_<your class>_<your name>.py

Task 3.3

Write a procedure `display_all` that will display all the information in the Rental collection. Output "NA" for size if the document does not contain size field. Include an additional column to display the total amount payable (i.e.: daily rental fee multiplied by the number of days rented).

You may refer to the following Python code to calculate the difference in the number of days between two dates:

```
from datetime import date
f_date = date(2014, 7, 2)
l_date = date(2014, 7, 11)
delta = l_date - f_date
print(delta.days)
```

Run the `display_all` procedure.

[8]

Add your program code to

TASK3_<your class>_<your name>.py

- 4 Style Theory's competitor JP Fashion uses a relational SQL database to manage its customers and rental details.

The table descriptions are as follow:

Customer(Email, FirstName, LastName, ContactNumber, DOB, Address)

Product(CatalogueNumber, Category, Brand, Size, Fee)

CustomerRental(ID, Email*, StartDate, EndDate)

ProductRental(ID*, CatalogueNumber*, Returned)

Note: * denotes foreign key

Task 4.1

Create an SQL file called `TASK4_1_<your class>_<your name>.sql` with the SQL code to create database `JPFashion.db` with the four tables.

[7]

Save your SQL code as

`TASK4_1_<your class>_<your name>.sql`

Task 4.2

The text files, `CUSTOMER.TXT`, `PRODUCT.TXT`, `CUSTOMERRENTAL.TXT` and `PRODUCTRENTAL.TXT` contain data for each table.

Write program code to read the records from the four text files, and insert all information from the files into the `JPFashion.db`.

[7]

Save your program code as

`TASK4_2_<your class>_<your name>.py`

Task 4.3

Customers may create a new account using JP Fashion's web application home page.

STEP 1: User to enter email address in a form.

STEP 2: Prompt user to re-enter email address if it exists in Customer table. Otherwise, get user to enter his/ her first name, last name, contact number, date of birth and address).

STEP 3: Display message "New account created successfully".

Write the code for a Python program as well as the necessary HTML files. Name the homepage `index.html`. And use 5678 as the port number.

[12]

Save your program code as

`TASK4_3_<your class>_<your name>.py`

Task 4.4

Staff may visit `rentalsdue.html` to filter the rental information by end date and display the results on the web browser.

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

- receives end date from a HTML form, then
- creates and returns a HTML document that enables the web browser to display the rentals due that day and have not been returned yet.

The output should include the following columns:

- Rental ID,
- catalogue number,
- email, and
- contact number.

Run the web application and input end date as `'2022-08-09'` in the webpage. [10]

Save the output of the program as

`TASK4_4_<your class>_<your name>.html`

END OF PAPER