YISHUN INNOVA JUNIOR COLLEGE
JC 2 PRELIMINARY EXAMINATION
Higher 2

CANDIDATE NAME			
CG		INDEX NO	

COMPUTING **9569/02**

Paper 2 (Lab-based)

31 Aug 2021
3 hours
100 Marks

Additional Materials: Removable storage device with the following files:

- `template.ipynb`
- `PEOPLE.csv`
- `POSRECORDS.txt`
- `Q3 Web App Folder`
- `DATA.txt`

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.

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 **13** printed pages and **1** blank page.

Instruction to candidates:

Your program code and output for Tasks 1, 2 and 4 should be saved in a single `.ipynb` and downloaded as

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

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:

In [2]:   #Task 1.2
          Program Code

          Output:

In [3]:   #Task 1.3
          Program Code

          Output:
```

1 During a contact tracing exercise, the TraceTogether token's tag number, the user's name and the user's mobile number are stored in an Abstract Data Type (ADT) `Person`. The information are stored as a three element tuple as follows:

```
(tag: INTEGER, name: STRING, hp: STRING)
```

Function	Return Type	Description
`make_person(tag,name,hp)`	`Person ADT`	A Constructor to create the `Person` ADT.
`get_tag(Person)`	`INTEGER`	Returns the tag number of the user.
`get_name(Person)`	`STRING`	Returns the name of the user.
`get_hp(Person)`	`STRING`	Returns the mobile number of the user.

Task 1.1

Write program code to implement the ADT `Person` with the above constructor and accessors. [4]

Task 1.2

Write program code for the function `read_file()` to read all the 21 users' information from the file `PEOPLE.csv` and return a list containing all the `Person` ADTs. [5]

Task 1.3

Write program code for the function `insertion_sort(lst)` that takes the list `lst` obtained from **Task 1.2** and sort them according to their tag numbers in an ascending order using the insertion sort algorithm. [5]

Searching for a tag number in a large list of users may be slow and tedious. However, if the list is sorted, performing a binary search would be more efficient.

Task 1.4

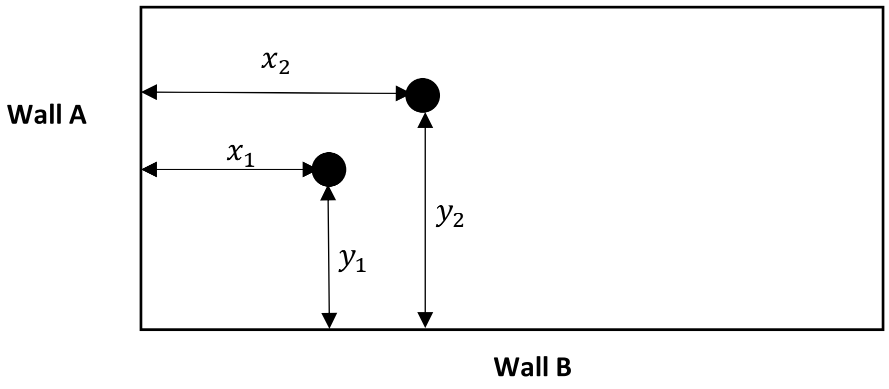
Write program code for the function `search(lst,num)` that searches for a tag number `num` in the sorted list `lst` obtained from **Task 1.3**. The code should

- perform a binary search on the sorted list.
- return the `Person` ADT if the tag number `num` exists. Otherwise, return `None`.
- print the number of comparisons made in the searching process. [8]

2 Every user is required to carry a tracing token inside an indoor sports facility so that the sensing device system can detect the token to read the position of the users and their temperature. The data is stored as records in the file `POSRECORDS.txt`, with the following entries separated by commas:

- the tag number (`INTEGER`) of the tracing token which could be used to identify the user
- the temperature (`FLOAT`) of the user measured in degree Celsius
- the location `(x,y)` (`FLOAT,FLOAT`) of the user which consists of the perpendicular `x`– and `y`– distances, measured in metres, from the walls A and B respectively

The diagram below shows two locations (x_1,y_1) and (x_2,y_2):



The distance between two locations can be calculated using the formula:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Task 2.1

Create an empty array `all_records` of size 20 and paste the data from the text file `POSRECORDS.txt` into your program code.

Write program code for the function `distance(user1,user2)` that takes in two records and returns the distance between the two users in metres, correct to 2 decimal places. [3]

For safety reasons during the Covid-19 pandemic, two users are considered to be in close proximity if the distance between them is less than 1.5 m apart.

Task 2.2

Write program code for the function `analyse(user1)` that iterates through all the records in the array `all_records`, calculates the distances between `user1` and all other users in the array, and returns a list of tag numbers of the users who are in close proximity to `user1`. [5]

Under the Safe Management Measures, people with a temperature of more than 37.5 degree Celsius will be flagged out as RED cases and the people who are in close proximity to these RED cases will be flagged out as YELLOW cases. The facility manager needs to submit both lists to the authority daily for follow-up actions.

Task 2.3

Write program code for the function `red_list(all_records)` that iterates through all the records in the array `all_records` and returns a list of tag numbers belonging to the RED cases. [2]

Task 2.4

Write program code for the function `yellow_list(red_cases,all_records)` that iterates through all the records in the array `all_records` to check for people who were in close proximity to any of the RED cases. The function returns a list of tag numbers belonging to the YELLOW cases.

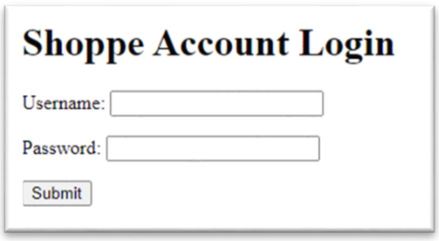
Those people flagged out as RED cases in **Task 2.3** should not appear in the list of YELLOW cases even though they may be in close proximity to another RED case.

You may use the functions written in **Task 2.1** and **Task 2.2** for the program code in this task. [5]

3 Shoppe e-Commerce has a mobile application for customers to make purchases through its online platform. All the customers' details, product data and ordering records are kept in the database `shoppe.db`.

Task 3.1

Write program code for the webpage `index.html` for the customer to log in to their account. The `/login` route in the server code should verify the customer's username and password using the data in the `Account` table in the database.



Shoppe Account Login

Username:

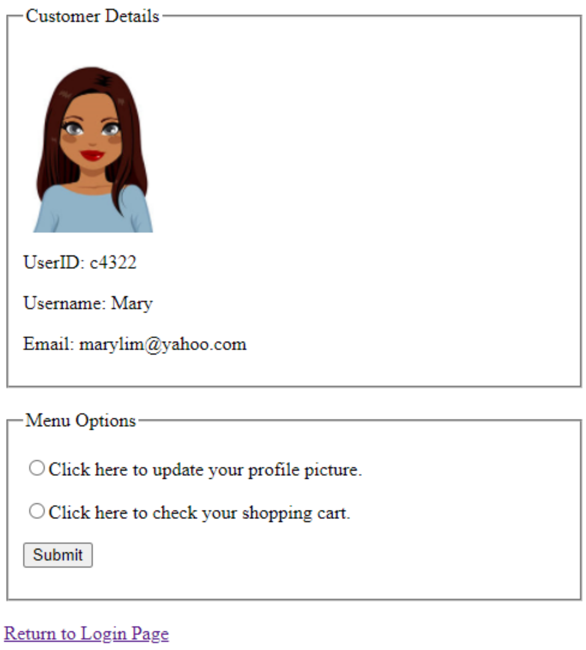
Password:

Submit

If the log-in details are valid, the customer will receive the webpage `display.html`. Otherwise, the customer will be redirected back to the log-in page. [7]

Task 3.2

Write program code for the webpage `display.html` to display the customer's details, with the profile picture, and a menu for the customer to choose the option to update the profile picture or to check the shopping cart.



Customer Details

UserID: c4322

Username: Mary

Email: marylim@yahoo.com

Menu Options

○ Click here to update your profile picture.

○ Click here to check your shopping cart.

Submit

Return to Login Page

[6]

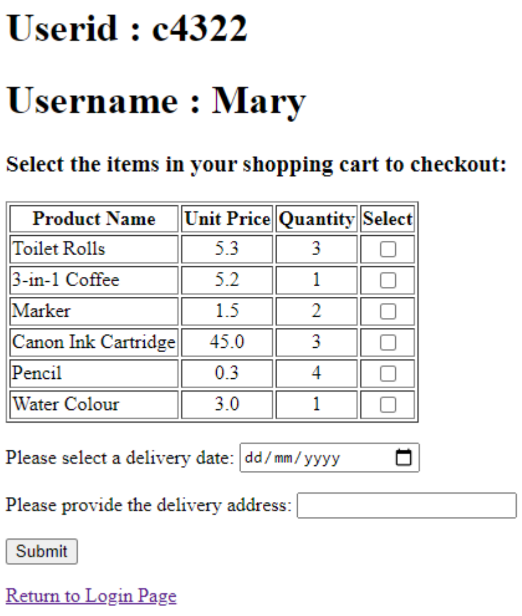
A customer, John, does not have a profile picture and he now wishes to upload the file `mypic.png`. This picture file will be renamed as `John.png` before storing in the web server.

Task 3.3

Write program code for the customer to select and upload a picture file, and it should include the following:

- `/menu` route in the server code to provide the customer with a webpage `profile.html` when the option to update the profile picture is chosen
- `profile.html` to allow the customer to upload a profile picture in the `.png` format
- `/update` route in the server code to receive the uploaded picture file and rename it as `<username>.png` before storing in the server's `\static\photo\` directory
- `success.html` to display a webpage informing the customer that the profile picture has been successfully uploaded [7]

The Shoppe customers usually browse through the available products on the platform and add them to their shopping carts. When they have decided on their purchase, they will select some or all the items in the shopping cart before checking out to make payment.



Userid : c4322

Username : Mary

Select the items in your shopping cart to checkout:

Product Name	Unit Price	Quantity	Select
Toilet Rolls	5.3	3	☐
3-in-1 Coffee	5.2	1	☐
Marker	1.5	2	☐
Canon Ink Cartridge	45.0	3	☐
Pencil	0.3	4	☐
Water Colour	3.0	1	☐

Please select a delivery date: dd/mm/yyyy

Please provide the delivery address:

Task 3.4

Write program code for the customer to select the items to check out for payment, and it should include the following:

- `/menu` route in the server code to query the `Cart` table in the `shoppe.db` when the customer chooses the option to check the shopping cart
- `cart.html` to display the list of items in the shopping cart and let the customer select them for checking out; the customer will also be required to indicate the preferred date and address for delivery
- `/checkout` route in the server code to receive the customer's inputs and insert a record into the `Orders` table in the `shoppe.db`
- `success.html` to display the **total cost** and inform the customer that the purchase has been successfully recorded. [14]

4 Lessonology is a learning management system that utilises gamification elements to motivate students to complete their assignments. The Linked List data structure is used to store the students' names and their total experience points. Each node contains a student's name, the student's total experience points, and a pointer to the next node. The nodes are linked together according to the order provided in the `DATA.txt` file.

A program is to be written to implement nodes as an instance of the class `Node`. The class `Node` has the following properties and method:

Class: `Node`	
Properties	
Identifier	**Description**
`Name`	The node's value for a student's name.
`Exp`	The node's value for the student's total experience points.
`Pointer`	The pointer to the next node.
Method	
Identifier	**Description**
`SetPointer()`	Set the pointer to point at the next node or point to `None` when it is the last node.

A linked list is implemented as an instance of the class `StudentList`. The class `StudentList` has the following property and methods:

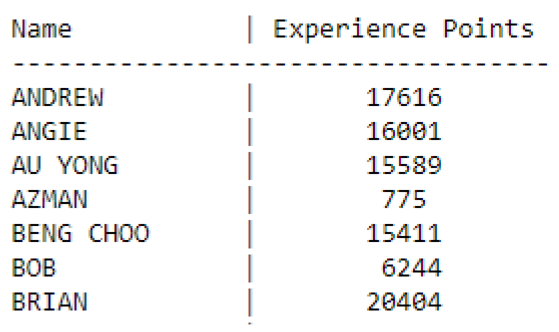
Class: `StudentList`	
Property	
Identifier	**Description**
`Start`	The pointer at the start of the linked list.
Methods	
`Constructor`	Initialise the linked list with the pointer `Start` assigned to `None.`
`Add()`	Add a new node into the linked list.
`Update()`	Update the value for the total experience points of a student's node in the linked list.
`Delete()`	Delete a node in the linked list.
`Display()`	Display the current content of the linked list in table form.

Task 4.1

Write program code for the classes `Node` and `StudentList`, including the `Constructor`, `Add()` and `Display()` methods. The code should follow the specification given. Do not write the `Update()` and `Delete()` methods yet.

The `Add(node)` method for the `StudentList` class should add the `node` containing a student's name and the student's total experience points to the linked list, according to the order given in the `DATA.txt` file.

Test your code by reading the data from the file `DATA.txt` and adding them as nodes into the linked list. The diagram below shows a portion of the expected output when using the `Display()` method on the populated linked list:



```
Name            | Experience Points
----------------------------------
ANDREW          |       17616
ANGIE           |       16001
AU YONG         |       15589
AZMAN           |         775
BENG CHOO       |       15411
BOB             |        6244
BRIAN           |       20404
```

[9]

Task 4.2

Each time a student completes an assignment, points will be awarded and the student's total experience points will be updated.

Write program code for the `Update(name,points)` method for the `StudentList` class that takes a student's `name` and the awarded `points` as inputs to update the student's total experience points in the node. (You may assume that the node containing the student exists in the linked list.)

For example, `Update('BRIAN',100)` will update the total experience points of a student whose name is `'BRIAN'` from `20404` to `20504`. [3]

Task 4.3

Write program code to implement the `Delete(name)` method for the `StudentList` class to search and remove a node, containing a particular student's `name`, in the linked list. Return `True` if the node is found and removed; otherwise return `False`. (You may assume that the students' names are unique in the linked list.) [4]

Task 4.4

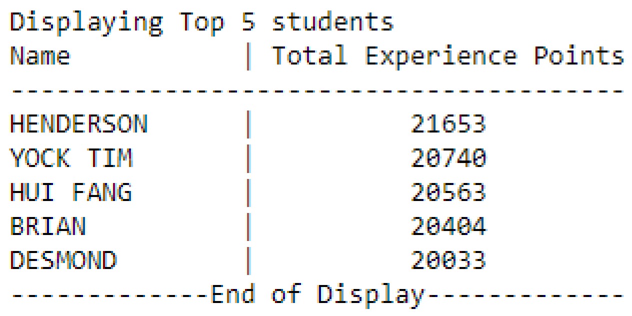
Another linked list which has pointers linking the nodes in decreasing order of the experience points is implemented as an instance of the class `Leaderboard`.

The class `Leaderboard` has the following properties and methods:

Class: `Leaderboard`	
Property	
Identifier	**Description**
`Start`	The pointer at the start of the linked list.
Methods	
`Constructor`	Inherit the property and all the methods from the class `StudentList`. Initialise the linked list with the pointer `Start` assigned to `None`.
`Add()`	Modify the `Add()` method in the parent class to add a new node in decreasing order of total experience points.
`Update()`	Modify the `Update()` method in the parent class such that the linked list is still in decreasing order of experience points after updating a student's total experience points.
`DisplayTop()`	Display the content of the nodes in the linked list for the top students based on their total experience points.

Write program code for the class `Leaderboard` to inherit the properties and methods from the class `StudentList` with the modified `Add()` and `Update()` methods. The additional `DisplayTop(n)` method should display the top `n` number of students in the linked list, based on their total experience points. (You may assume that no two students have the same total experience points.)

Test your code by reading the data from the file `DATA.txt` and adding them as nodes into this linked list. The diagram below shows the expected output when using the `DisplayTop(5)` method on the linked list:



```
Displaying Top 5 students
Name           | Total Experience Points
----------------------------------------
HENDERSON      |         21653
YOCK TIM       |         20740
HUI FANG       |         20563
BRIAN          |         20404
DESMOND        |         20033
-------------End of Display-------------
```

[13]

BLANK PAGE