Name:___________________________________ **Class:**___________

JURONG PIONEER JUNIOR COLLEGE
JC 1 Year – End Examination 2022

COMPUTING **9569/02**

Higher 2 **22 September 2022**

Paper 2 **2 hours**

Additional materials: Cover Page
Electronic version of `TASK2_DATA.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.

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 **7** printed pages

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 The vehicle registration number consist of a front section with alphabets, followed by a middle section with numerals, and ending with an alphabet.
Some examples are:

```
S 1 Y
EG 10 A
SJP 9999 G
```

The following are the criteria for the format of the vehicle number:

1. First character begins with ‘S’ or ‘E’
2. Front characters are alphabets of length 1 to 3
3. Middle characters correspond to a number between 1 to 9999 (inclusive)
4. Last character is an alphabet

For this question:

- **single-spacing** is used between the front characters and the numbers, and between the numbers and the last character, and
- assume that the alphabets in the test data are all in uppercase.

Task 1.1

Write program code for a function `veh_num_check(string)` that takes in a `string` and returns integer **`0`** if it is a valid vehicle number. Otherwise, the function returns integer **`1`**, **`2`**, **`3`**, or **`4`** for their invalidity corresponding to failing the criteria given above. Each input has at most one type of invalidity. [8]

Task 1.2

Call your function `veh_num_check` with suitable inputs for **five test cases**, which includes one test case for each return value described above. Each input has at most one type of invalidity.

For every test case, put in comments using hash #, the expected return value and the explanation of the test. [5]

Download your program code for Task 1 as

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

2 1000 randomly generated integers between 1 to 10000 (inclusive) are stored in a text file `TASK2_DATA.txt`.

Task 2.1

Write a function `read_from_file(filename)` that:

- takes the name of the text file as input argument,
- reads the data from the file, and
- returns the data as a list of integers.

Call your function `read_from_file` with `TASK2_DATA.txt`, printing the returned list and its length, using the following statements:

```
list1 = read_from_file(“TASK2_DATA.txt”)
print(list1)
print(len(list1))
```

[4]

Task 2.2

Write a function `quick_sort(list_of_integers)` that:

- takes a list of integers,
- implements a quick sort algorithm, and
- returns the sorted list of integers.

Call your function `quick_sort` with the data of the file `TASK2_DATA.txt`, printing the returned list, using the following statements:

```
list1 = read_from_file(“TASK2_DATA.txt”)
list1_sorted = quick_sort(list1)
print(list1_sorted)
```

[6]

Task 2.3

Write a function `linear_search(list_of_integers, target)` that:

- takes a **sorted** list of integers,
- implements the linear search algorithm to search for `target`, without making any redundant comparison in the search algorithm, and
- returns `True` if `target` is found, and `False` otherwise.

Call your function `linear_search` by passing in a sorted list of integers. Assign `2022` to `target`, printing the returned list. Use the following statements:

```
list1 = read_from_file(“TASK2_DATA.txt”)
list1_sorted = quick_sort(list1)
print(linear_search(list1_sorted, 2022))
```

[4]

Task 2.4

Write a function `binary_search(sorted_list, target)` that:

- takes a **sorted** list of integers
- implements the binary search algorithm to search for target
- returns `True` if target is found, else return `False`

Call your function `binary_search` by passing in a sorted list of integers sorted using the `quick_sort` function from **Task 2.2**, and target is 2022, printing the returned list. For example, using the following statements:

```
list1 = read_from_file('TASK2_DATA.txt')
list1_sorted = quick_sort(list1)
print(binary_search(list1_sorted, 2022))
```

[5]

Download your program code for Task 2 as

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

3 A Certificate of Entitlement (COE) gives the right to own and use a vehicle in Singapore.

<u>How the COE Bidding System Works</u>
The reserve price is the maximum bid amount that a bidder is prepared to pay. Bidders outbid each other to obtain a COE during the bidding exercise by keying in their reserve price.

Task 3.1
A COE bid consists of customer ID, reserve price, and status.

```
                 COEBid
-------------------------------------------
cust_id: STRING
reserve_price: INTEGER
status: STRING
-------------------------------------------
constructor(id:STRING, price:INTEGER)
get_cust_id(): STRING
get_reserve_price(): INTEGER
get_status(): STRING
set_status(s: STRING)
display()
```

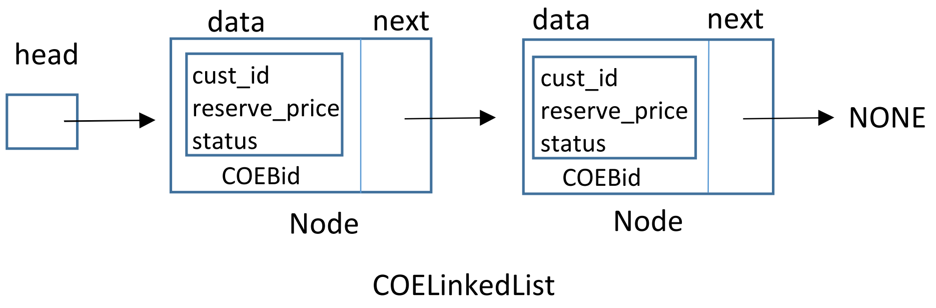
The `constructor` initialises customer ID and reserve price according to the parameters, and the status to the value “`Bidding in progress`”.

The `display()` method prints the customer ID, reserve price and status.
Sample output: `T001, $41, Bidding in progress`

Write the `COEBid` class according to the class diagram above. [6]

Task 3.2
The bids in the COE Bidding System can be maintained by a linked list. The linked list uses a node implementation.

Each node has a `data` attribute that references a `COEBid` object, and a `next` attribute that points to the next node.



Node
`data: COEBid` // `COEBid` object `next: Node` // `Node` object
`constructor(data: COEBid)` // initialises `next` as NONE

COELinkedList
`head: Node` // first `Node` object

`constructor()`
`insert_bid(data: COEBid)`
`display_list()`
`calculate_COEPrice(quota: INTEGER): INTEGER`
`update_status(quota: INTEGER)`

Bids are sequenced in a LinkedList in **descending order** of their reserve price when customer make their bids.

- The `insert(data:COEBid)`method will insert the `COEBid` object into the LinkedList in **descending order** of the reserve price.

- The `display_list()`method will display the information of all the COE bids stored in the linked list in **descending order** of the reserve price.

Write the program code to define the following:
- `Node` class, and
- `COELinkedList` class (with constructor, `insert` and `display_list` methods). [10]

Test your program by writing the codes to do the following:
- instantiate a `COELinkedList` object called `COE_LL`,
- instantiate five `COEBid` objects using the information in the following table and insert them in `COE_LL`, using the `insert_bid` method, and

`cust_id`	`reserve_price`
T001	41
T002	88
T003	67
T004	70
T005	100

- print the information of all the `COEBids` stored in `COE_LL` using the `display_list` method. They should appear in descending order of the reserve price. [3]

Task 3.3

<u>CALCULATION OF COE Price</u>

If the COE quota is **three** and the number of bidders is **five** with reserve prices of:

S$100, S$88, S$70, S$67 and S$41.

The COE Price is the reserve price of the first unsuccessful bidder + $1. That is, the COE Price would be S$(67 + 1) = S$68.

Reserve Price	**Bid Status**	**Remarks**
$100	Successful	First 3 bids will be successful as the COE quota is 3. The COE Price will be S$68.
$88	Successful	
$70	Successful	
$67	Unsuccessful	The 4th and 5th bids are unsuccessful based on the COE quota of 3.
$41	Unsuccessful	

- The `calculate_COEPrice(quota:INTEGER)`method will calculate the COE price according to the quota following the description above and return the COE price.
- The `update_status(quota:INTEGER)`method will iterate through the bids in the linked list, and according to the value of quota, update status of the `COEBid` objects to ‘`Successful`’ or ‘`Unsuccessful`’ by calling the `set_status` method.

Write the `calculate_COEPrice` and `update_status` methods in the LinkedList class.

Test your program with the following code:

```
quota = 3
COE_price = COE_LL.calculate_COEPrice(quota)
print(“The COE price is”, COE_price)
COE_LL.update_status(quota)
COE_LL.display_list()
```

[9]

Download your program code for Task 3 as

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

END OF PAPER