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 Jurong Pioneer Primary School uses buses to transport students to school. There are six bus routes labelled A to F. A survey was conducted to analyse the punctuality statistics of these buses over a four-week period.

The data from the survey is stored in the file `SURVEY.TXT`. The format of the data in the file is:

```
<Day>,<A>,<B>,<C>,<D>,<E>,<F>
```

Positive numbers represent minutes early, negative numbers represent minutes late and `0` represents the bus having been on time.

Task 1.1

Write the program code that:

- reads the entire contents of `SURVEY.TXT` into an appropriate data structure called `Records`, and
- displays the contents of `Records` in neat columns. [4]

Task 1.2

Extend your program so that the following statistics for the four-week period may be calculated and output:

- the number of late arrivals for each bus route,
- the average number of minutes late for each bus route, using only data from days on which it was late, and
- the bus route(s) with the highest number of days late.

All the results should be displayed with appropriate annotation. The following is an example run of the program:

```
                                        A      B      C      D      E      F


No. of late arrivals for each bus route 3      0      1      2      1      0

Average no. of minutes late             1.5    0.0    1.0    2.0    1.0    0.0


Bus route A has the highest number of days late.
```

[8]

Task 1.3

Additional code is to be written for the user to input a specific day, for example: `Fri3`, to be used for the analysis of data. Find and display how many buses were late on this day and for each late bus, display the route label and how late the bus was on this day.

Test your code using the following test data:

`Tue1`

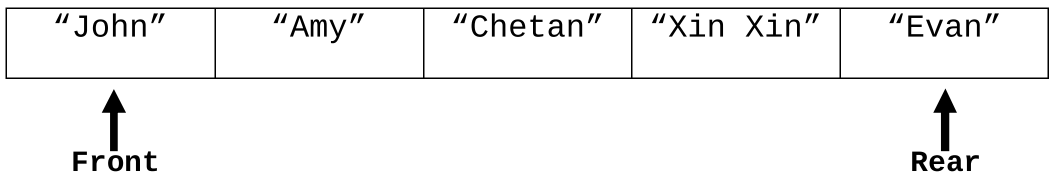
`Thu2`

Download your program code for Task 1 as

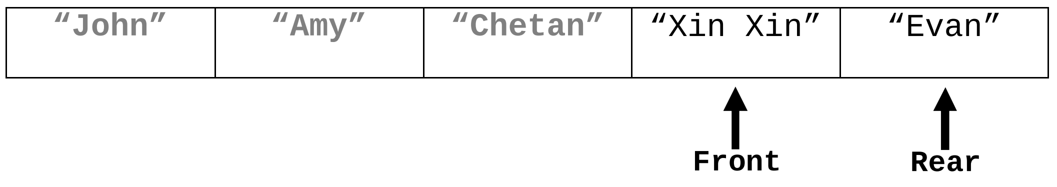
`TASK1_<your class>_<your name>.ipynb` [4]

2 In linear queue data structure, elements are inserted until the queue becomes full. However, after the queue becomes full, new elements cannot be inserted until all the existing elements are removed from the queue. Although there are empty spaces in the queue, they remain unused. This is a disadvantage of a linear queue.

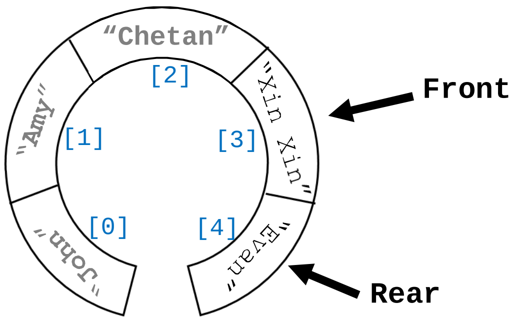
After inserting all the elements into a linear queue:



Linear queue is still considered full after elements have been dequeued:



To overcome this disadvantage, a circular queue data structure may be implemented. The next element added to the queue will be stored at index 0



Queue

Attributes

Identifier	Data Type	Description
`Items`	`ARRAY[0:4] OF STRING`	Stores the elements of queue.
`Front`	`INTEGER`	Index of the first item added to the queue.
`Rear`	`INTEGER`	Index of the last item added to the queue.

Methods

Identifier	Description
`Constructor()`	Instantiates a `Queue` object.
`IsEmpty()`	Returns `TRUE` if the queue is empty and `FALSE` otherwise.
`IsFull()`	Returns `TRUE` if the queue is full and `FALSE` otherwise.
`Enqueue(STRING)`	Inserts a new item to the queue. Displays a suitable message if the queue is full.
`Dequeue():STRING`	Returns the item removed from the queue or `“NONE”` if the queue is empty.
`Display()`	Outputs items from the front to the rear of the queue.

CircularQueue

Methods

Identifier	Description
`Constructor()`	Instantiates a `CircularQueue` object.
`IsFull()`	Returns `TRUE` if the queue is full and `FALSE` otherwise. Overrides the method in parent class.
`Enqueue(STRING)`	Inserts a new item to the queue. Displays a suitable message if the queue is full. Overrides the method in parent class.
`Dequeue():STRING`	Returns the item removed from the queue or `“NONE”` if the queue is empty. Overrides the method in parent class.
`Display()`	Outputs items from the front to the rear of the queue. Overrides the method in parent class.

Task 2.1

Implement the classes `Queue` and `CircularQueue` with object-oriented programming. The first item added to an empty queue is stored at index 0. The attributes of each object is reinitialised when the queue becomes empty. [20]

Task 2.2

There are two printers in the General Office. One of the printers implements a linear queue while the other implements a circular queue.

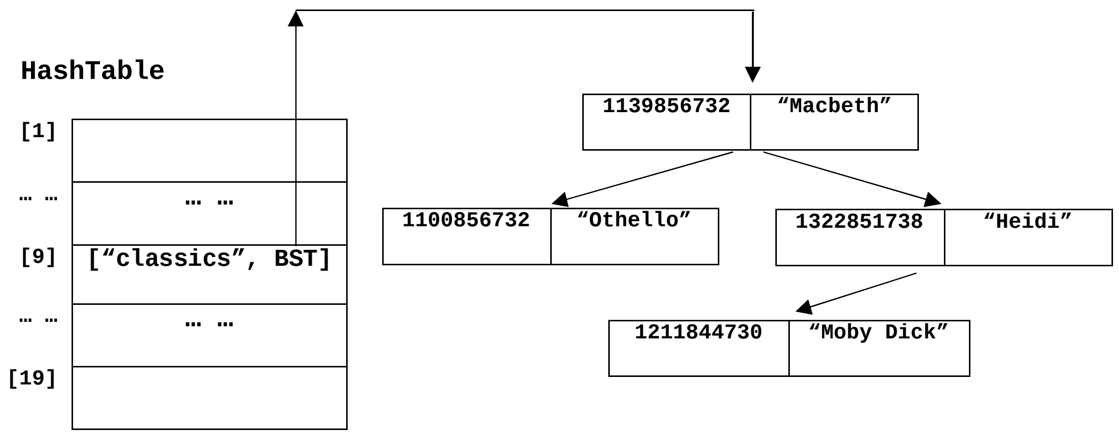
Write the code to instantiate a `Queue` object and a `CircularQueue` object. Test your code, on **both queues**, using the following steps:

i. Enqueue five users in the order given in the diagram.
ii. Dequeue twice.
iii. Enqueue `“Mohan”`.
iv. Display the queue.

Download your program code for Task 2 as

`TASK2_<your class>_<your name>.ipynb` [6]

3 The library in Jurong Pioneer Primary School uses a hybrid data structure to keep track of its inventory. Each record in the hash table stores a simple list `[<Category>, <Binary Search Tree object>]`. Each node in the Binary Search Tree (BST) stores the ISBN number and title of a book. The nodes in each BST share the same book category and are sorted, in ascending order, according to their ISBN numbers.



BSTNode
- ISBN: INTEGER - Title: STRING - Left: BSTNode - Right: BSTNode

+ Constructor(INTEGER, STRING)
+ GetISBN(): INTEGER
+ SetISBN(INTEGER)
+ GetTitle(): STRING
+ SetTitle(STRING)
+ GetLeft(): BSTNode
+ SetLeft(BSTNode)
+ GetRight(): BSTNode
+ SetRight(BSTNode)

BST
- Root: BSTNode

+ Constructor()
+ GetRoot(): BSTNode
+ SetRoot(BSTNode)
+ AddNode(INTEGER, STRING)
+ Display() //uses in-order traversal

- Private
+ Public

Task 3.1

Write the object-oriented code for the `BST` and `BSTNode` classes described above. [10]

[Turn over

A checksum is applied to determine the Hash Value for each `<Category>`, where the ASCII value of each character in the title is multiplied by its position in the `<Category>` string (starting from left to right), and then summed.

For example, given the category `“classics”`, the summed value would thus be: 99×1 + 108×2 + 97×3 + 115×4 + 115×5 + 105×6 + 99×7 + 115×8 = 3884

(3884 `MOD` 19) + 1 = 9

A weighted modulus 19 operation is then applied and 1 is added to the remainder to determine the final Hash Value.

Task 3.2

Write the code for the function `CalcHash(my_string)`, which takes in a string argument and returns its resultant Hash Value. [4]

Task 3.3

Write the code to declare and initialise `HashTable`, an empty hash table array that may store up to 19 records. [2]

Task 3.4

`CATEGORIES.TXT` is a text file containing the book categories. Read the entire contents of `CATEGORIES.TXT` and update the records in the hash table. Collisions are handled using **linear probing**. [4]

`BOOKS.TXT` holds the details of books in the library. The format of the data in the file is: `<Category>,<ISBN>,<Title>`.

Task 3.5

Write program code to:

- read the lines from the file,
- extract the `<Category>`, `<ISBN>` and `<Title>` values, and
- add each book to the `BST` of its category. [6]

Task 3.6

Write the code to display the ISBN and title of each book belonging to `“classics”` category. The output is sorted according to the ISBN numbers. Ensure your output uses headings to identify the data displayed.

Download your program code for Task 3 as

`TASK3_<your class>_<your name>.ipynb` [4]

4 A computer company has several offices throughout Singapore, each with several salespersons. Each salesperson is assigned to one office only. A record of the sales made by each salesperson has been set up using a relational database.

The following tables hold the data:

```
CUSTOMER (CustomerID, CustomerName, Email, Telephone)
OFFICE (OfficeID, PostalCode, Telephone)
SALE (SalesPersonID*, CustomerID*, SaleDate, Amount)
SALESPERSON (SalesPersonID, SalesPersonName, OfficeID*)
```

Note: Underlined field indicates primary key. Asterisk (*) indicates a foreign key.

Task 4.1
Write the SQL code to create the four tables in the database named `computercompany.db`.

Save the SQL file as `TASK4_1_<your class>_<your name>.sql`. [4]

Task 4.2
The files `CUSTOMER.CSV`, `OFFICE.CSV`, `SALE.CSV` and `SALESPERSON.CSV` contain information exported by their spreadsheets files. Write Python code to migrate them to the database.

Save your Python code as `TASK4_2_<your class>_<your name>.py`. [6]

Task 4.3
Write SQL code to show the `SaleDate`, `SalesPersonName`, `CustomerName` and `Amount` of all sale transactions performed at the office with ID `1`.

Save the SQL file as `TASK4_3_<your class>_<your name>.sql`. [4]

Task 4.4
A report is produced to show the top salesperson in each office each month. Write Python code to:

i. generate a web form that allows a user to enter the month (1 to 12) and year,
ii. use the data submitted by the web form to query the database, and
iii. return a HTML page with the report displayed.

The following is a sample report for March, 2020.

Top Performers in March 2020

Office ID	Salesperson	Total Amount S$
1	Low Kok Kheong	7880
2	Mindy Tan	6935
3	Monish Chandra	10700

[Turn over

Save your Python program as `TASK4_4_<your class>_<your name>.py` with any additional files/ sub-folders as needed in a zipped folder named `Task4_<your class>_<your name>.zip`. [12]

Task 4.5

Deploy the web app on your local host and enter the following data:

- month – `8`, and
- year – `2020`.

Save the screenshot of table generated as `TASK4_5_<your class>_<your name>.jpg`. [2]

END OF PAPER