NANYANG JUNIOR COLLEGE
JC2 PRELIMINARY EXAMINATION

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **26 August 2020**

3 hours

Additional Materials: Electronic version of `waste.csv` data file
Electronic version of `story.csv` data file
Insert Quick Reference Guide

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.

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 question or part question. The total number of marks for this paper is 100.

Instruction to candidates:

Your program code and output for each of Task 1 to 3 should be saved in a single `.ipynb` file. For example, your program code and output for Task 1 should be saved as
`TASK1_<your name>.ipynb`

1 The task is to implement a hash table to retrieve data about waste disposal in Singapore.

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:
```

The file `waste.csv` contains the following fields in each line:

```
Year – "YYYY"
Waste Disposed Of – "Numeric" (Million Tons)
Waste Recycled – "Numeric" (Million Tons)
```

The first line of the file contains the headings.

Task 1.1

Write a program to:

- read data from `waste.csv`
- into a hash table of size 20
- by creating a function `GetKeyAddress(Year)` to generate the hash address
- and directly inserting the data into the correct location in the hash table
- taking care of any potential collisions using any suitable methods

Display the contents of the hash table showing the data from the first slot to the last slot.

[14]

Task 1.2

Create a menu with the following options:

```
1. Get Waste Disposed and Recycled by year
2. Display year(s) where Recycled waste > Waste disposed
3. Return Average waste disposed between two years
4. -1 to Exit
```

- implement the functions for each menu choice.
- use only direct access to retrieve data for options 1 and 3.
- option 1 and 3 requires asking users to input in the year(s).
- validate the user input.
- test option 1 with the year 2007 and show the output.
- test option 3 with the range 2002 to 2008 and show the output.
- show the output for option 2.

[10]

Save your program code and output for Task 1 as
`TASK1_<your name>.ipynb`

2 A school library stores the following data in a file named `story.csv`:

Field	Format
book_title	text
subject	text
author_name	text
published	'YYYY' (year)

Merge sort is an efficient sorting algorithm which falls under divide and conquer paradigm and produces a stable sort. It operates by dividing a large array into two smaller subarrays and then recursively sorting the subarrays.

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 2.1
        Program code

        Output:

In [2]: #Task 2.2
        Program code

        Output:
```

Task 2.1

Write program code to:

- read data from `story.csv` into an array of records.
- ask user to input in which field to sort the records by.
- validate that the choice must be `'B'`, `'S'`, `'A'`, or `'P'` representing `book_title`, `subject`, `author_name` and `published` fields.
- implement a `MergeSort(ArrayData, Sortby)` function that takes in two parameters, `ArrayData` (array of records) and `Sortby`, and sorts the records in ascending order according to the specified field. `MergeSort(ArrayData, Sortby)` will return the sorted `ArrayData` using a mergesort algorithm to do the sorting.
- display `ArrayData`.
- test your program twice and show your output for sorting by `subject` and by `author_name`.

[12]

Task 2.2

Write program code to:

- implement a `QuickSort(ArrayData)` function that uses the quicksort algorithm to sort the `ArrayData` by `book_title` in descending order.

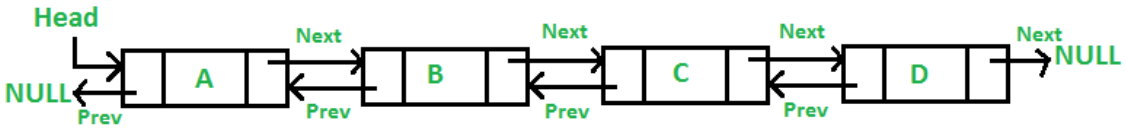
[8]

Design 2 test cases to test your `QuickSort(ArrayData)` function and explain the purpose of the test data. Show the output of your test cases.

[4]

Save your program code and output for Task 2 as
`TASK2_<your name>.ipynb`

3 You are to create a song playlist using a doubly linked list implemented using Object-Oriented Programming (OOP). The doubly linked list data structure is a linked list made up of nodes with two pointers pointing to the next and previous element.



A doubly linked list allows traversal of nodes in both direction which is not possible in a singly linked list. For example, a user can go forward or backwards to play the next or previous song.

The `node`, will store the following data:

- `title` (`str`)
- `prev` (`node`)
- `next` (`node`)

The class, `SongPlaylist`, will store the following data:

- a doubly linked list of `node` objects.
- `head` pointer, pointing to the first `node` in the doubly linked list.

The class `SongPlaylist` has the following methods:

- `insert(SongPlaylist, title: str)` adds a song title at the beginning of the list.
- `insertafter(SongPlaylist, searchtitle: str, newtitle: str)` adds a song title after `searchtitle.`
- `display(SongPlaylist)` outputs out the playlist.

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 3.1
         Program code
```

```
In [2]:  #Task 3.2
         Program code
```

```
In [3]:  #Task 3.3
         Program code
```

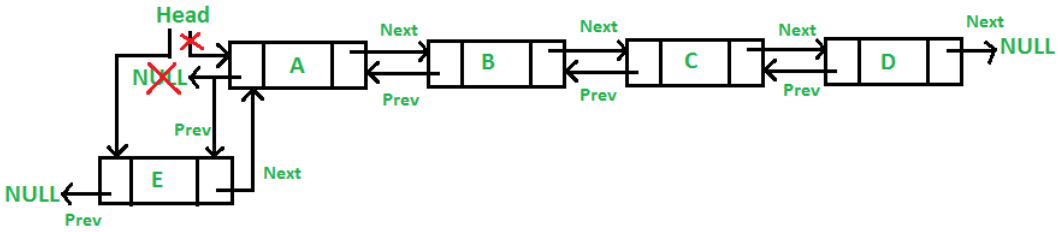
```
Output:
```

Task 3.1

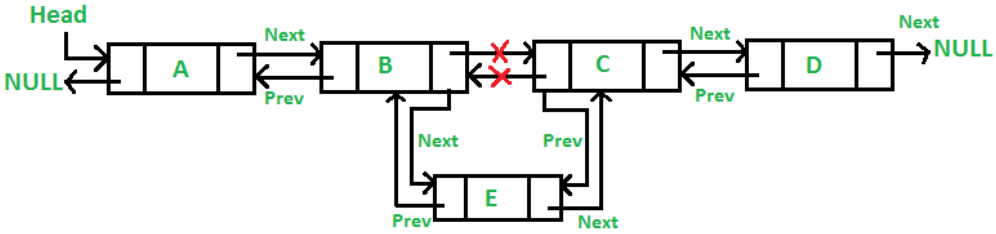
Write program code to define classes to implement the song playlist.

The figures below show the links that must be updated for the `insert` and `insertafter` methods:

Inserting at the beginning of the list



Inserting after a given node



[10]

The program has the following menu:

```
1. Create New Song Playlist
2. Add a song in front
3. Add a song after
4. Display Playlist
5. -1 to End
```

Option 1 will prompt the user to enter the name of the new playlist.
Option 2 will prompt the user to enter the song title and the name of the playlist.
Option 3 will prompt the user to enter the name of the playlist, existing song title, and the new song title to be inserted after the existing song title.
Option 4 will prompt the user to enter the name of the playlist to be displayed and output accordingly.

Task 3.2

Write program code to:

- display the main menu.
- input the choice by the user.
- run the appropriate code to carry out the task for the choice made.

[4]

Test your program by creating a new playlist called `MJ` and add in the following titles in order: `"Heal the World", "Thriller", "Beat It"`. Show your output by displaying the playlist.

[2]

Task 3.3

Extend your program by writing a function `insertionSort(Playlist)` that will sort the song titles in ascending order.

The algorithm for this `insertionSort` is given below:

1) Create an empty `sorted` doubly linked list.
2) Traverse the given doubly linked list, do following for every node.
   - Insert current node in sorted way in `sorted` doubly linked list.
3) Change head of given linked list to head of `sorted` list.

Write program code to implement `insertionSort(Playlist)`.

[6]

Save your program code and output for Task 3 as
`TASK3_<your name>.ipynb`

4 A school wants to create a database to allow students to register for different enrichment activities that will be held on the school's Enrichment Day. An enrichment activity falls under one of three categories – Arts, Cultural, and Sports.

It is expected that the database should be normalised.

When a student registers for an activity, the following information is recorded:

- `StudentID` - unique 6-digit register number of the student.
- `Type` - type of activity ('`A`', '`C`' or '`S`').
- `Venue` - where the activity will be held.
- `Session` - whether the activity is conducted in the morning or afternoon (`'AM'` means the morning session, and `'PM'` means the afternoon session).

For the Arts category, the following extra information is recorded:

- `Performance` – "`True`" for performance arts, "`False`" for visual arts.

For Cultural, the following extra information is recorded:

- `Race` – which race the culture belongs to.

For Sports, the following extra information is recorded:

- `Contact` – "C" to denote contact sports, "NC" to denote non-contact sports.
- `Cost` - the amount of money in dollars (not more than $50) the student must pay the instructor.

The information is to be stored in four different tables:

```
Registration
Arts
Cultural
Sports
```

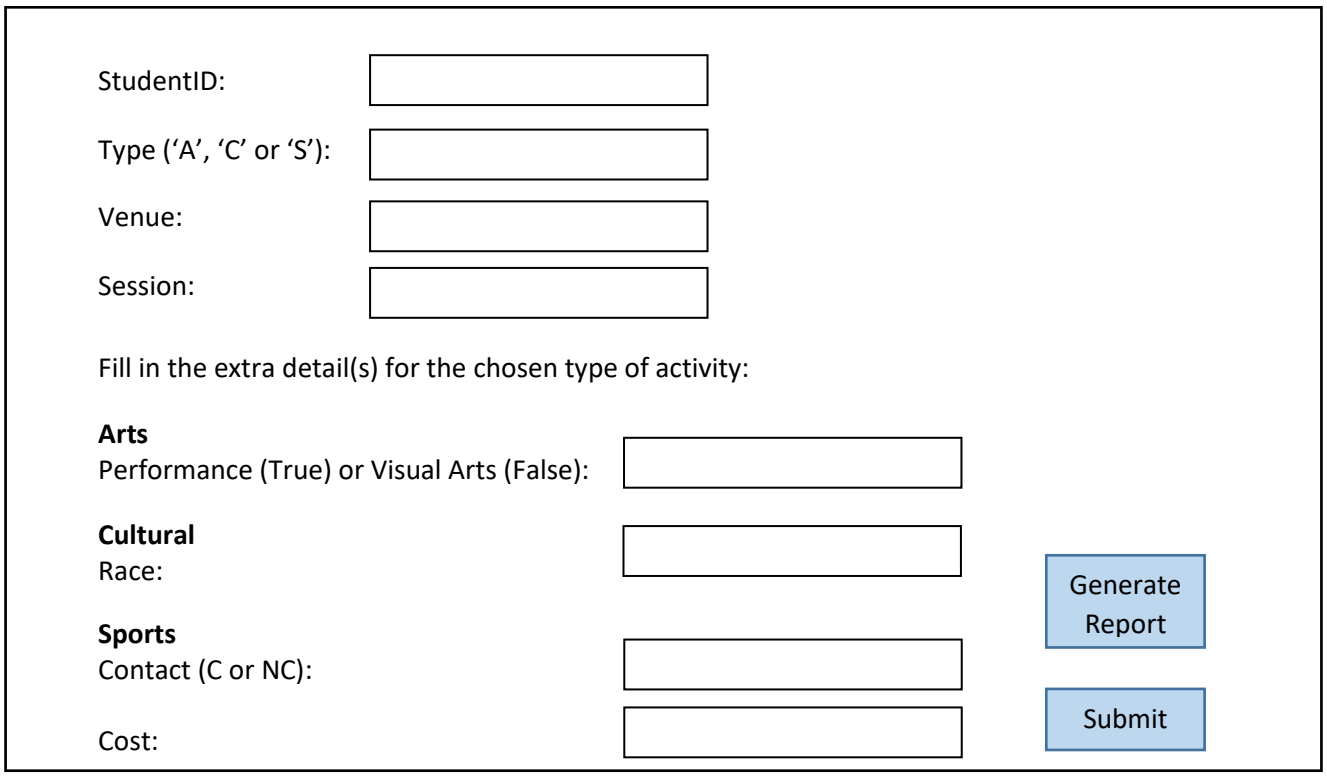
Task 4.1

Create an SQL file called `TASK4_1_<your name>.sql` to show the SQL code to create the database `register.db` with the four tables. The table, `Registration`, must use `StudentID` as its **primary key.** The other tables must refer to the `StudentID` as a **foreign key**.

Save your SQL code as
`TASK4_1_<your name>.sql` [5]

Task 4.2

The school wants to allow students to register over the internet. The form design for the webpage is shown below:



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

- accepts the input from the web form (assume input is keyed in correctly)
- updates the registration details into `register.db`
- creates and returns a HTML document that enables the web browser to display a table tabulating the `StudentID` and `Type` of activity registered for the morning session.

Save your Python program as
`TASK4_2_<your name>.py` with any additional files / sub-folders as needed in a folder named `TASK4_2_<your name>`

[12]

Task 4.3

Test your web application by entering the following records via the form's submit button:

```
192701, A, Hall, AM, True
192703, A, MPR, PM, False
192723, S, Field, AM, C, 20
192803, C, 5-56, AM, Malay
192820, S, 5-60, PM, NC, 15
193005, C, LT4, PM, Chinese
193006, C, LT4, PM, Chinese
```

Save the output of the program when the user clicks on the "Generate Report" button as `TASK4_3_<your name>.html`

[3]

Task 4.4

Write SQL code to count the number of different races for the cultural activities. Run this query.

Save this code as
`TASK4_4_<your name>.sql`

[4]