NYJC 2020 MYE P2Q2 [25 marks]

1. 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.

   In the recursive approach, the problem is broken down into smaller, simple subproblems in **top-down** manner until the solution becomes trivial.

   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:
```

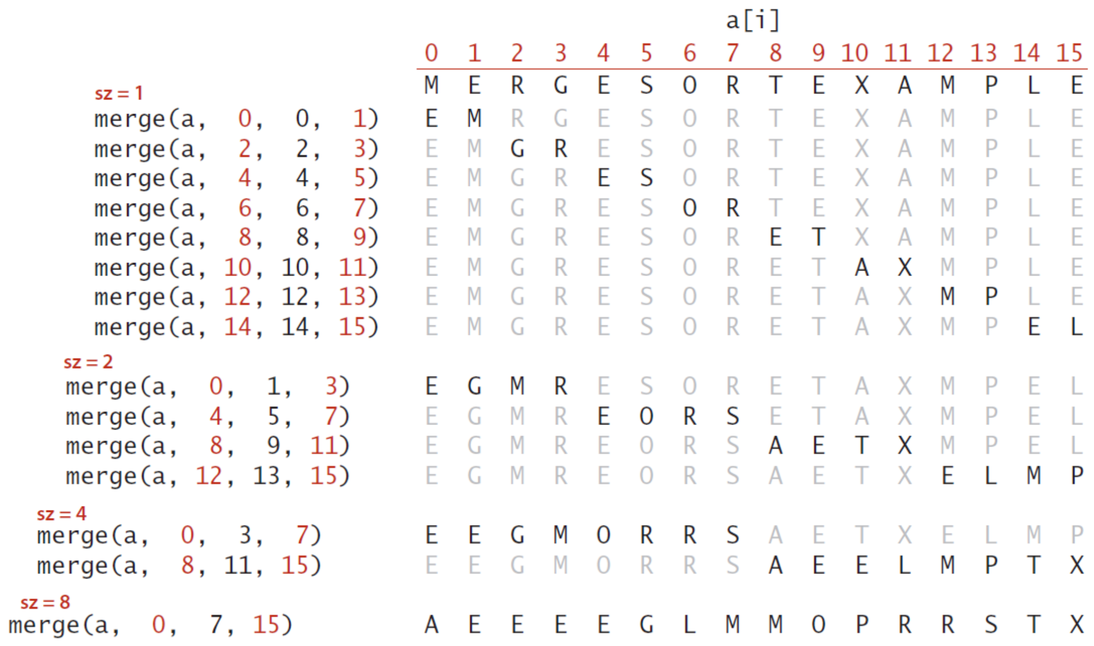
Task 2.1

Write program code to:

- create an array of 50 random integers between 0 - 99.
- display these numbers in rows of 10 integers.
- implement the function MergeSort(SortArray) that takes in an array of unsorted integers and sort them in ascending order.
- call MergeSort() function to sort your array of 50 random integers.
- display the sorted result in rows of 10 integers.

[12]

Bottom-up mergesort (non-recursive) consists of a sequence of passes over the whole array, doing merges on subarrays of size sz, starting with sz equal to 1 and doubling sz on each pass. The final subarray will be of size sz if the array size is an even multiple of sz, otherwise it is less than sz.



```
                                                a[i]
                              0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
                              M  E  R  G  E  S  O  R  T  E  X  A  M  P  L  E
       sz = 1
       merge(a,  0,  0,  1)   E  M  R  G  E  S  O  R  T  E  X  A  M  P  L  E
       merge(a,  2,  2,  3)   E  M  G  R  E  S  O  R  T  E  X  A  M  P  L  E
       merge(a,  4,  4,  5)   E  M  G  R  E  S  O  R  T  E  X  A  M  P  L  E
       merge(a,  6,  6,  7)   E  M  G  R  E  S  O  R  T  E  X  A  M  P  L  E
       merge(a,  8,  8,  9)   E  M  G  R  E  S  O  R  E  T  X  A  M  P  L  E
       merge(a, 10, 10, 11)   E  M  G  R  E  S  O  R  E  T  A  X  M  P  L  E
       merge(a, 12, 12, 13)   E  M  G  R  E  S  O  R  E  T  A  X  M  P  L  E
       merge(a, 14, 14, 15)   E  M  G  R  E  S  O  R  E  T  A  X  M  P  E  L
     sz = 2
     merge(a,  0,  1,  3)     E  G  M  R  E  S  O  R  E  T  A  X  M  P  E  L
     merge(a,  4,  5,  7)     E  G  M  R  E  O  R  S  E  T  A  X  M  P  E  L
     merge(a,  8,  9, 11)     E  G  M  R  E  O  R  S  A  E  T  X  M  P  E  L
     merge(a, 12, 13, 15)     E  G  M  R  E  O  R  S  A  E  T  X  E  L  M  P
   sz = 4
   merge(a,  0,  3,  7)       E  E  G  M  O  R  R  S  A  E  T  X  E  L  M  P
   merge(a,  8, 11, 15)       E  E  G  M  O  R  R  S  A  E  E  L  M  P  T  X
 sz = 8
 merge(a,  0,  7, 15)         A  E  E  E  E  G  L  M  M  O  P  R  R  S  T  X
```

Trace of merge results for bottom-up mergesort

Task 2.2

Write program code to:

- implement a non-recursive iterative MergesortLoop(Array, SortOrder) function to sort an array of 50 integers.
- allow the user to specify whether to sort in ascending (`'A'`) or descending order (`'D'`).
- display the sorted result in rows of 10 integers.

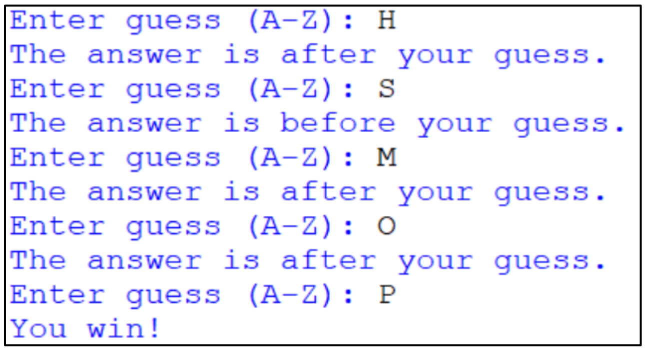
[10]

Design 2 test cases to test your function and explain the purpose of the test data. Show the output of your test cases.

[3]
Save your program code and output for Task 2 as
`TASK2_<your name>.ipynb`

HCI 2020 P2Q3 [10 marks]

1. Design the server program and client program for an asymmetric guess-the-letter game where a server generates a random capital letter from A to Z and a client tries to guess it. After each incorrect guess, the server returns whether the answer is before or after the guessed letter. Once the client guess the letter correctly, both the client and server quit the game. Below is a sample run of the client.



```
Enter guess (A-Z): H
The answer is after your guess.
Enter guess (A-Z): S
The answer is before your guess.
Enter guess (A-Z): M
The answer is after your guess.
Enter guess (A-Z): O
The answer is after your guess.
Enter guess (A-Z): P
You win!
```

The socket protocol is to use `\n` to detect the end of a message and you do not need to handle invalid input of client. [10]

Save your program code as
`TASK3_client_<your name>_<ct>.py` and
`TASK3_server_<your name>_<ct>.py`

Run the program. Produce a screenshot of the output and save it as
`TASK3_<your name>_<ct>.jpg`

PJC 2016 P1Q4 [35 marks]

4. A college uses a binary tree structure to store its subjects offered to students.

The program will use a user-defined type **`Node`** for each node defined as follows:

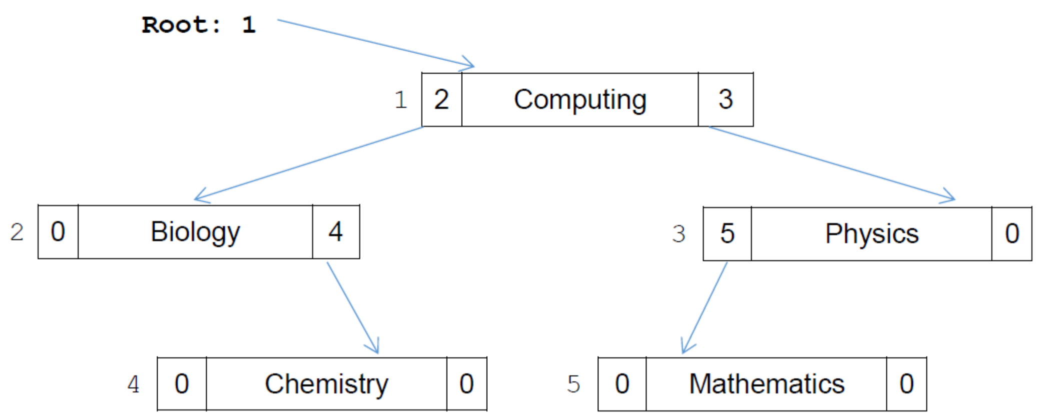
Identifier	Data type	Description
`Subject`	`STRING`	The node's value for subject offered
`LeftPtr`	`INTEGER`	The left pointer for the node
`RightPtr`	`INTEGER`	The right pointer for the node

A linked list is maintained of all unused nodes which do not form part of the tree. The first available node which is used for a new item is indicated by **`NextFreePosition`**. Items in the unused list are linked using their left pointers.

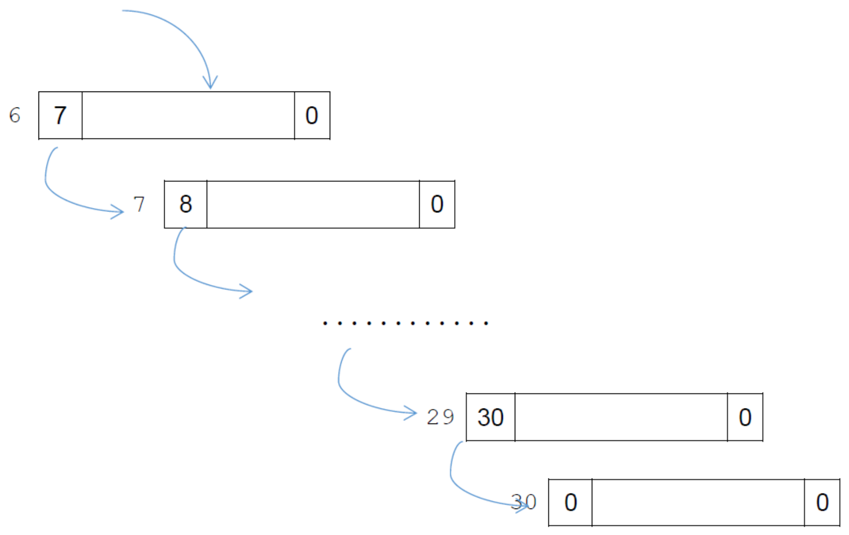
The binary tree and linked list are implemented using variables as follows:

Identifier	Data type	Description
`SubjectTree`	`ARRAY[30] of Node`	The tree data
`Root`	`INTEGER`	Index position of the root node
`NextFreePosition`	`INTEGER`	Index for the next unused node

The diagram shows the binary tree and linked list after five subjects have been added.



`NextFreePosition: 6`



Task 4.1

Write the program code to declare all the required variables and create the initial linked list which contains all 30 nodes.

Add statement(s) to initialise the empty tree.

Evidence 16: Your program code for task 4.1. **[8]**

The following incomplete pseudocode inserts a data value into the binary tree structure.

```
PROCEDURE InsertBinaryTree(NewItem)
    ... ...
    IF tree is empty
        THEN
            ... ...
        ELSE
            //traverse the tree to find the insert position
            ... ...
            LastMove = 'X'
            REPEAT
                PreviousPtr ← CurrentPtr
                IF NewItem < CurrentPtr item
                    THEN
                        //move left
                        CurrentPtr ← CurrentPtr's left
pointer
                        LastMove = 'Left'
                    ELSE
                        //move right
                        CurrentPtr ← CurrentPtr's right
pointer
                        LastMove = 'Right'
                ENDIF
            UNTIL CurrentPtr = NULL
    ENDIF
    IF LastMove = 'Left'
    ... ...
    ELSE IF LastMove = 'Right'
    ... ...
    ENDIF
    ... ...
END PROCEDURE
```

Task 4.2

Complete the pseudocode and write a module `AddSubject` to add a subject into the binary tree.

Evidence 17: Your `AddSubject` program code. **[7]**

Task 4.3

Write a module `Display` to display the value of `Root`, the value of `NextFreePosition` and the contents of `SubjectTree` in index order.

Evidence 18: Your `Display` program code. **[4]**

Task 4.4

Write a module `BuildTree` to construct a binary tree using the data provided in the data file `SUBJECT.TXT`. Read in all the data from the data file and use the `AddSubject` module.

Evidence 19: Your `BuildTree` program code. **[2]**

Evidence 20: Run your program and produce screenshot of contents of binary tree. **[1]**

Deleting a node from a tree may change the structure of the tree. To simplify the deletion process, label a node as "**`deleted`**" but do not remove the node from the tree structure. After that, regenerate the entire tree structure.

Task 4.5

Write a module `LabelDelete` that labels a node as "**`deleted`**" but **do not remove** the node from the tree structure.

Evidence 21: Your `LabelDelete` program code. **[6]**

Task 4.6

Write program code to regenerate the entire binary tree.

Evidence 22: Your program code to regenerate the binary tree. **[6]**

Evidence 23: Display a screenshot of the regenerated binary tree after deleting "`Chemistry`" and "`History`". **[1]**

RVHS 2020 CT3 P2Q4 [34 marks]

Task 4 – Result Management System

The school would like to implement an online result management system to record results of various subjects for students. In this task, you are required to implement a prototype using normalised database and flask web application to manage these records.

The following information of each `Student` is stored:
`MatricNo` – unique string in the format of `"RVHS-YYYY-XXX"` where `YYYY` is the year of entry to school and `XXX` is a 3-digit string ranged from `"001"` to `"999"`.
`Name` – name of student
`Class` – class of student
`IndexNo` – index number of the student in the class
`Gender` – gender of student, to be stored as a single character, using either `"M"` or `"F"`

The following information of each `Test` is stored:
`TestID` – unique autoincrement integer to identify the tests
`Subject` – subject name of the test, e.g. `"Computing"`
`Level` – academic level of the test, e.g. `"JC1"`
`MaxScore` – maximum score of the test, e.g. `40`
`Year` – year when the test is held, e.g. `2020`
`Term` – term when the test is held, e.g. `2`
`Percentage` – an integer representing the percent weightage of the test for the whole year, e.g. `12` (which means `12%`)

The following information of each `Result` is stored:
`MatricNo` – matric number of the student
`TestID` – test id of the test
`Score` – the score of the student for this test

The information is to be stored in three tables:
`Student`
`Test`
`Result`

Task 4.1
Create an SQL file called `Task_4_1.sql` to show the SQL code to create the database `result_mgm.db` with the three tables.

The table `Student` must use `MatricNo` as its primary key, and the table `Test` must use `TestID` as its primary key. The table `Result` should use `MatricNo` and `TestID` as a composite key, while `MatricNo` and `TestID` must refer to `MatricNo` in `Student` and `TestID` in `Test` as foreign keys.

Save your SQL code as
`Task_4_1.sql`

[5]

Task 4.2

The files `students.csv`, `tests.csv` and `results.csv` contains information about the student, test and results. The first row of each file contains the header of the respective columns. Each row in the files is a comma-separated list of information.

Write a Python program to insert all information from the three files into the database, `result_mgm.db`. Run the program.

Save your program code as
`Task_4_2.py` [5]

Task 4.3

A teacher would like to generate the **Economics** results of class **19J08** for year **2020**. Query and display a list of data with the following fields as shown in the table, sorted in the **ascending** order according to **TestID** of the test, followed by **IndexNo** of the student.

Class	IndexNo	Name	TestID	Score	MaxScore
…	…	…	…	…	…

Write the SQL code required.

Save this code as
`Task_4_3.sql` [6]

Task 4.4

The school wants to implement a function where teachers can enter the year and class to display information of the students from this class.

Write a Python program and the necessary files to create a web application that:
- Receive the `Class` from a HTML form, then,
- Creates and returns a HTML document that enables the browser to display a table tabulating the students' information in 5 columns, `Class`, `IndexNo`, `Name`, `Gender` and `MatricNo`,
- The list of information should be sorted according to the `IndexNo` of students.

Save your program as
`Task_4_4.py`
With additional files or sub-folders as needed in a folder named
`Task_4_4`

Run the web application. Enter the following `class`:
Class: 19J08

Then save the output of the program as `Task_4_4.html.`

[12]

Task 4.5

Lastly, the table should also be formatted according to the following requirements:

1. Table should occupy the width of the entire html document.
2. Boarder of table and cells should be displayed.
3. `Class` and `Name` column to be aligned to the left.
4. `IndexNo` column to be aligned to the center.
5. The other columns to be aligned to the right.
6. The background of header row should be formatted using color code of `"#12164c"`, and text color to be set as `"white"`.
7. The background color of rows of female students should be set as `"#fe8c68"`.
8. The background color of rows of male students should be set as `"#68d4f8"`.

Make necessary adjustment to files in folder Task_4_4 to reflect the above formatting changes.

Run the web application. Enter the following `class`:
Class: 19J08

Then save the output of the program as `Task_4_5.html`.

[6]