ASSIGNMENT-6

Language - Java

Deadline :28th March, 2023 EOD

Evaluation: 29th March, 2023(10:00AM – 12:00PM)

Question 1:

Registration Form interface using Java.

NOTE:

You are NOT allowed to use the drag-drop option of IDE like Netbeans or Eclipse. You have to write code from scratch. We will know if you have done so.

You can ONLY USE Visual Studio Code or Notepad.

A college has decided to give a form to the students willing to take admission to its BTech course. The objective of the form is to register for the online test that the college will be taking. They want to design a User Interface that gives the data to the Database for direct insertion.

Create a Student Registration form using the Java Swing library; it will be a "*JFrame*" titled "**Registration Desk.**" The form needs to be connected to a back-end Database system. The form must have the following fields:-

a. **Name** which will be a "*JLabel*". Corresponding to it, there shall be a box for the user to write. The box will be a "*JTextField.*"

b. **DOB**, which should also be a "*JLabel*". Corresponding to the label, we shall have a "*JTextField*."

c. **Gender**, which is also a "*JLabel.*" Corresponding to the label, there shall be two "*JRadioButton*," namely *Male* and *Female.*

d. **Email**, which should also be a "*JLabel*". Corresponding to the label, we shall have a "*JTextField*.".

e. **Password**, which should also be a "*JLabel*". Corresponding to the label, we shall have a "*JTextField*."

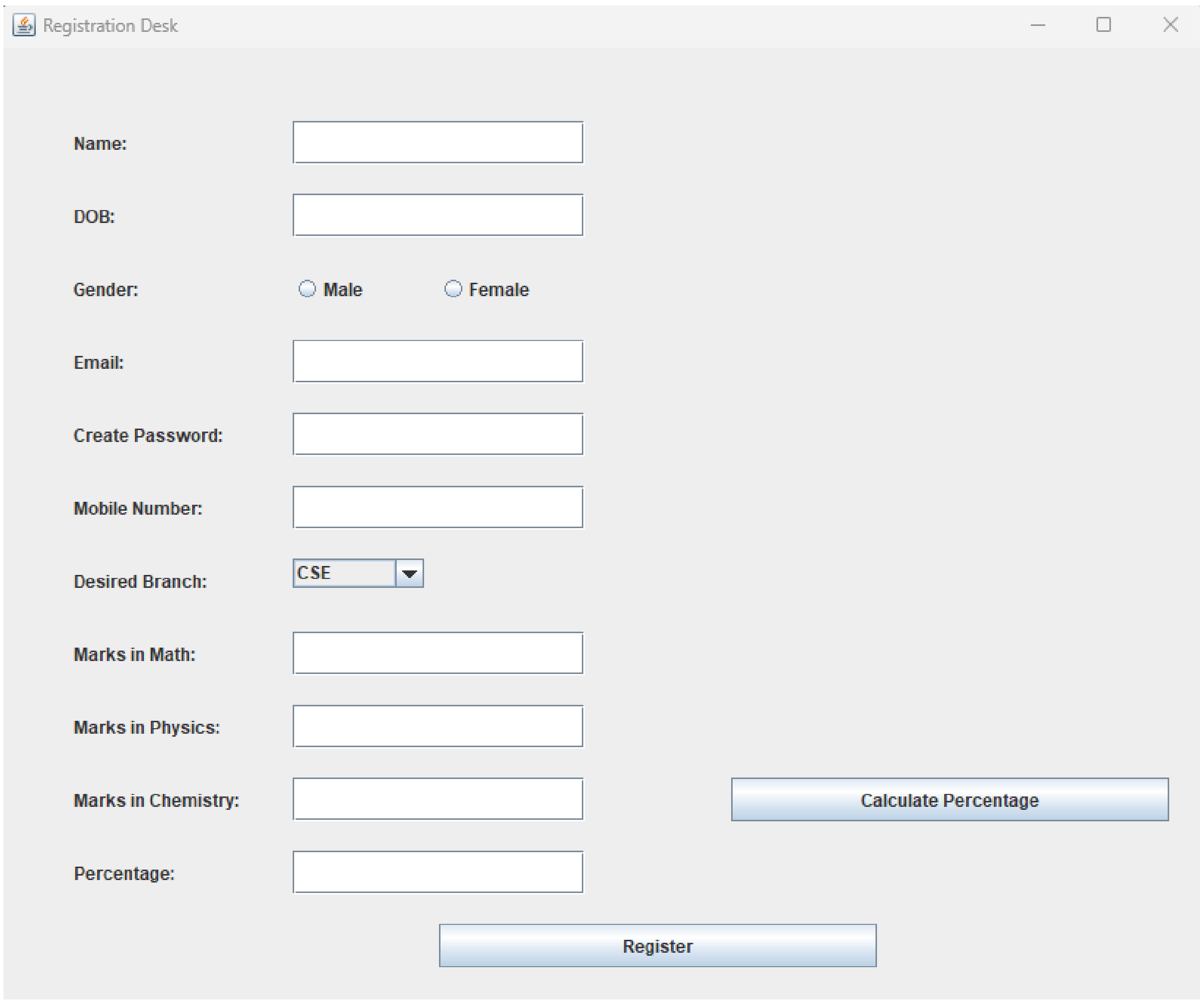
f. **Mobile No.**, which should also be a "*JLabel*". Corresponding to the label, we shall have a "*JTextField*."

g. **Desired Branch**, which should also be a "*JLabel*". Corresponding to the label, we shall have a drop-down menu "*JCombobox"* of 5 branches: CSE, *ME, EEE, ET, and CE.*

h. There should be three more "*JLabels*," **Marks in Maths**, **Marks in Physics**, *and* **Marks in Chemistry**. Corresponding to each label, a text field should be present, which is a "*JTextField."* Along with a small **Calculate percentage** "*JButton*." Once the calculate button is clicked, it should calculate the percentage and show it in the text field corresponding to the next label, “*Percentage*.” The percentage is calculated for physics, chemistry, and math and subsequently displayed.

Note: Maximum Marks is 100. No need for error checking here.

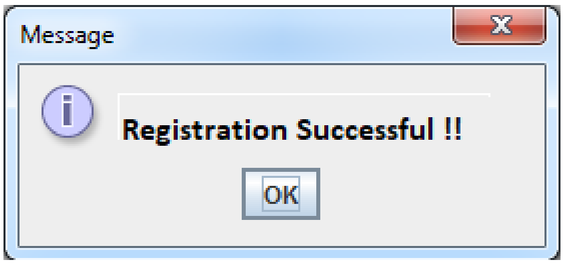
i. **Percentage** is a *Jlabel* with a “*JTextField*” corresponding to it, whose value should NOT be written by the User; instead, the value should be fetched as mentioned in point(h).
j. Finally, there should be a “*JButton,*” namely, Register, which, on clicking, must display a Message **Registration Successful**. For this, use "*JOptionPane*."



Registration Desk

Name:		
DOB:		
Gender:	○ Male ○ Female	
Email:		
Create Password:		
Mobile Number:		
Desired Branch:	CSE	
Marks in Math:		
Marks in Physics:		
Marks in Chemistry:		Calculate Percentage
Percentage:		
	Register	

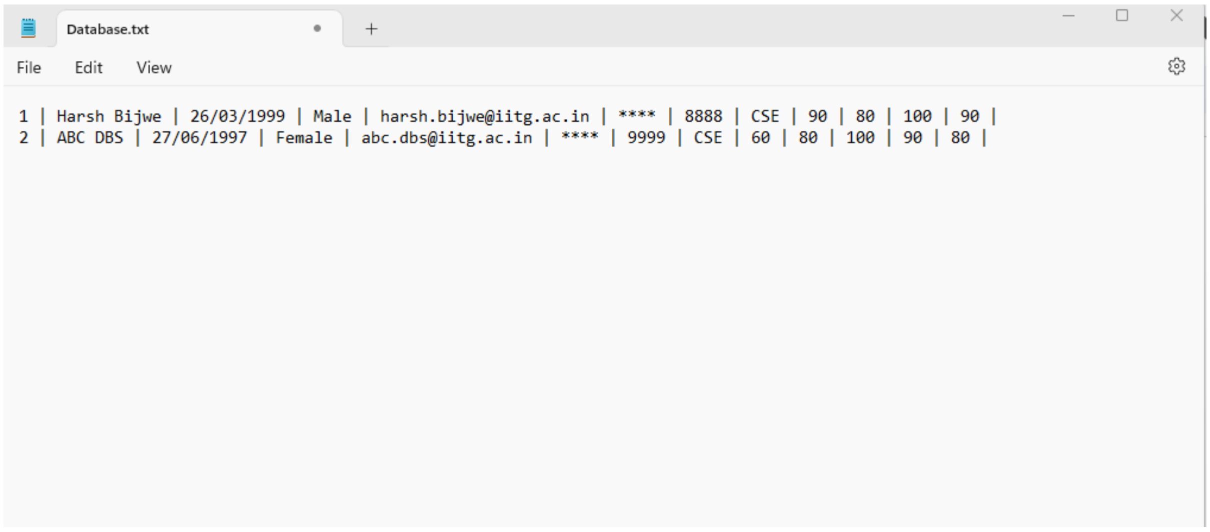
And on clicking Register, the following should be shown.


Message

Registration Successful !!

k. The form should reload, with all values flushed, to enter new values. This should be a continuous process until someone closes the *JFrame*.

l. Connect the User Interface to a database. Now for simplicity of the assignment, you may take a simple txt file (however, possibilities are endless, you may connect a MySQL Database or MS Excel as well, but for this assignment let's keep things simple) and store all the details of a registered user to that txt file in the following format. Name that file as "**Database.txt**". Each line must have all the details of a student separated by '|'.



```
1 | Harsh Bijwe | 26/03/1999 | Male | harsh.bijwe@iitg.ac.in | **** | 8888 | CSE | 90 | 80 | 100 | 90 |
2 | ABC DBS | 27/06/1997 | Female | abc.dbs@iitg.ac.in | **** | 9999 | CSE | 60 | 80 | 100 | 90 | 80 |
```

Note: For these Question, you will be required to implement LinkedList for Creating Adjacency List and Priority Queue or Efficient Sorting Algorithm for taking edges in Non-decreasing Order.

***** All the data structure you will have to implement you can't use any Library Functions. *****

Question: 2

Implement kruskal algorithm to find Minimum Spanning Tree (MST)

MST:

showAdjacencyList (): print adjacency list.

showGraph (): show the graph.

showMST() → In case of multiple MST print anyone.

showMSTInTheGraph() -> highlight edges those are part of MST in the graph and remaining edges with different colour.

kruskal() → implementation of algorithm

Input Example:

T

N E

Vi Vj W

…….

Test Cases: input will be given in the file which contains following details

T → No of test cases

N → No of Vertices (start from 0 to N-1)

E → No of Edges

W → weight

Example:

2

3 3

0 1 3

2 1 1

0 2 4

5 8

0 1 2

0 2 5

0 3 1

0 4 3

1 2 4

2 3 5

2 4 4

3 4 6

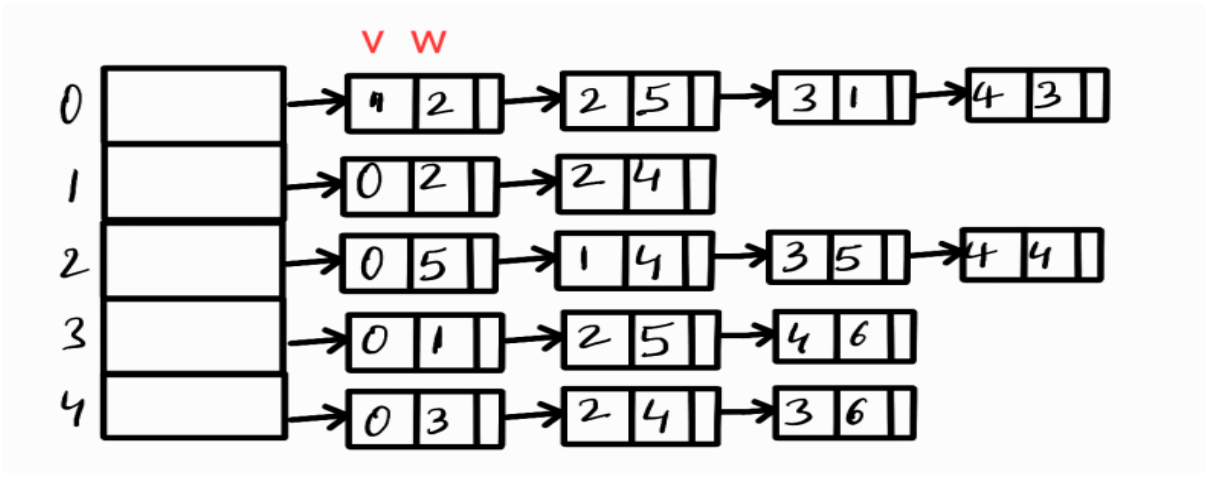
Output Format **(** Using GraphViz)**

Create an output file and generate .ps or .png file

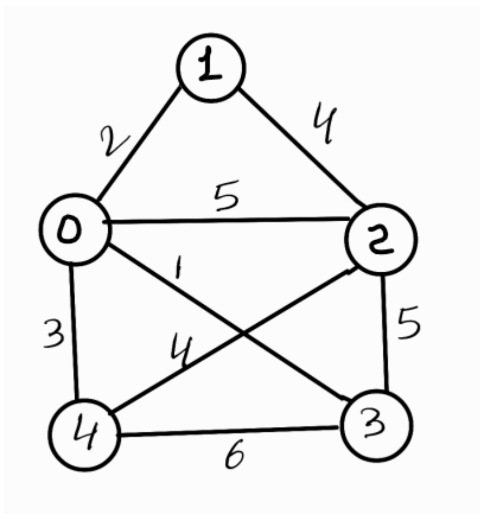
1. Print Adjacency List:
2. Print Graph
3. Print MST
4. Print MST in the Graph.

2nd Test case Expected Output.

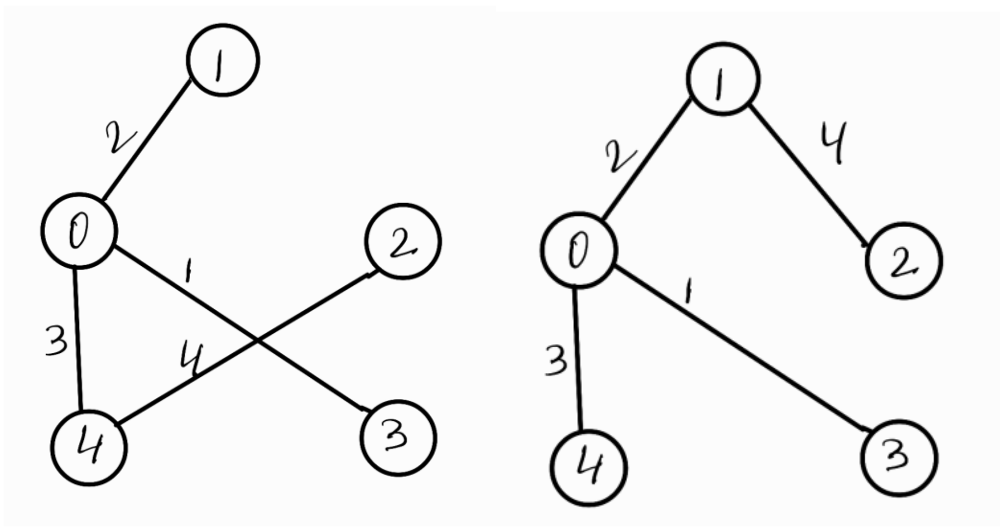
1)Adjacency List



2) Graph

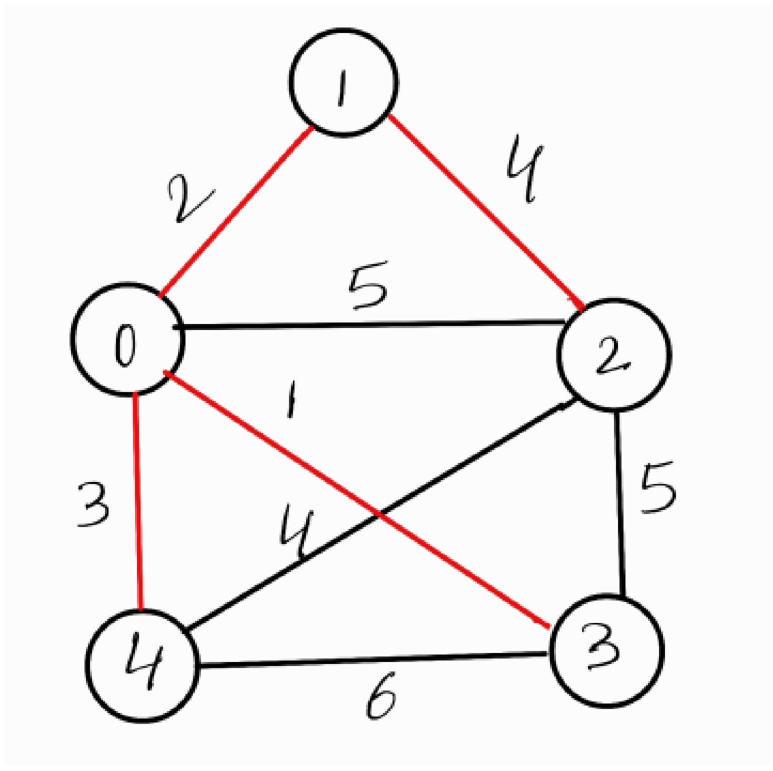


3) MST



4) MST in the Graph:

(In this graph you can see the MST edges highlighted in Red Colour)



Question: 3

Find Shortest paths between given source and destination in an undirected graph. (Dijsktra's algorithm)

Dijsktra's algorithm :

showAdjacencyList (): print adjacency list.

showGraph (): show the graph.

showShortestPath() → print the shortest path between source and destination.

(In case of multiple path print any one).

showShortestPathInTheGraph() -> highlight edges those are part of MST in the graph and remaining edges with different colour.

Dijsktra's () → implementation of algorithm

Input Example:

T

N E

Vi Vj W

…….

S D

Test Cases: input will be given in the file which contains following details

T → No of test cases

N → No of Vertices (start from 0 to N-1)

E → No of Edges

W → weight

S -> Source Vertex

D -> Destination

Example:

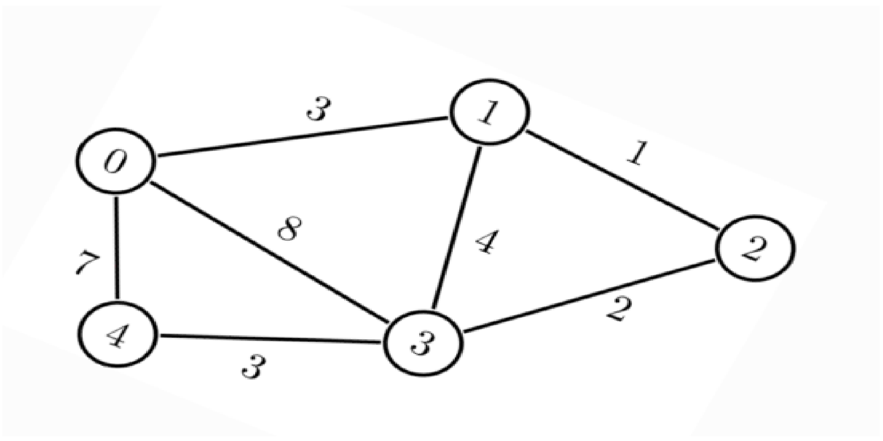
```
1
5 7
0 1 3
0 3 8
0 4 7
1 2 1
1 3 4
2 3 2
3 4 3
0 2
```

<u>Output Format</u> (****** Using GraphViz)

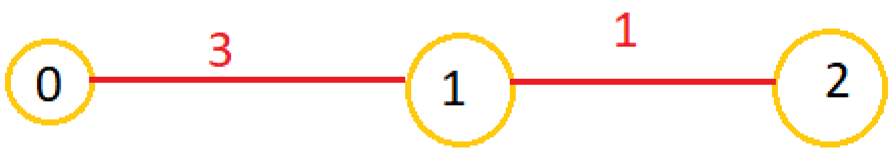
Create an output file and generate .ps or .png file

5. Print Adjacency List: (**same as MST**)
6. Print Graph
7. Print Shortest Path
8. Print Shortest Path in the Graph. → Highlight Vertices and Edges of shortest Path.

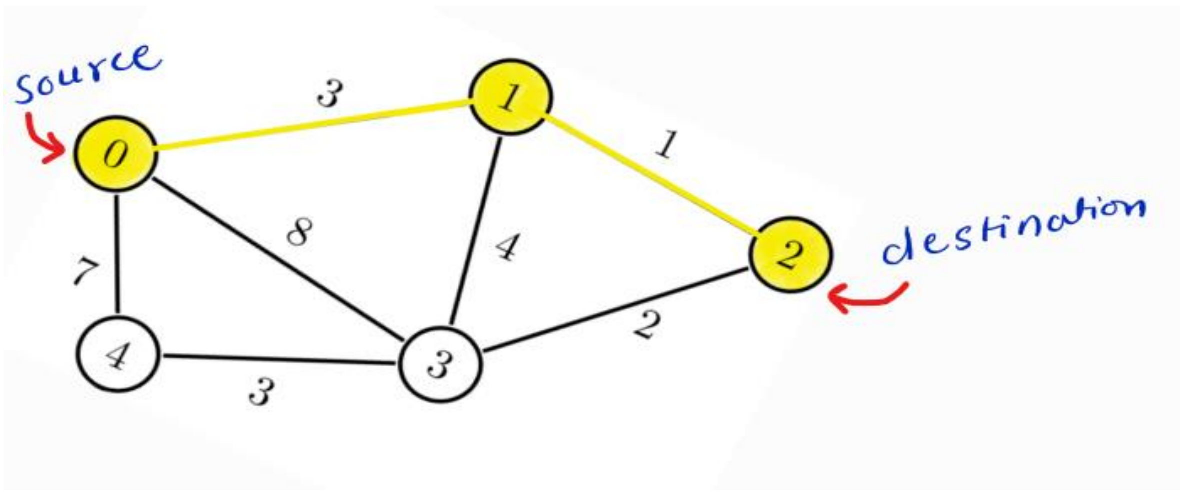
<u>Example:</u> **2)** Graph output



3) shortest Path output



4) Print Shortest Path in the Graph


source
destination
0
1
2
3
4
3
1
8
4
7
2
3