Sir Syed University of Engineering and Technology
Data Structures and Algorithm (SWE-203/CE-205/CS-212/CS-223)
Lab Quiz

Instructions: Each student must upload a file which contains the object-oriented code for the task. Also take screen shot of the output and attach it.

Note: For Batch 2015, 2016 and 2018

If your roll number ends at 0, 1, 2 you have to attempt Q1.

If your roll number ends at 3, 4, 5 you have to attempt Q2.

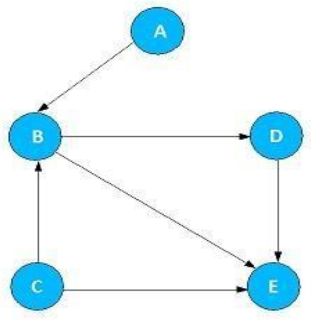
If your roll number ends at 6,7,8,9 you have to attempt Q3.

Q1. Create a class for a group of 50 students holding data for their Name, Roll number, CGPA, Section. The data should be collected by user input.

a. Print the roll number of first 10 students only.
b. Ask the user to starting roll number and ending roll number and print their CGPA
c. Apply linear search and print the roll number of the student who have highest CGPA.

Q2. Initialize a queue and enqueue the numbers in it as user input. Now split the queue into two sub queues — one for the front half, and one for the back half. If the number of elements is odd, the extra element should go in the front queue. So FrontBackSplit() on the list {2, 3, 5, 7, 11} should yield the two queues {2, 3, 5} and {7, 11}

Q3. Implement the following graph. Ask the user source node and destination node of the path and apply Depth First Search (DFS) to source to destination.



Note: For Batch 2017 and 2019

If your roll number ends at 0, 1, 2 you have to attempt Q1.

If your roll number ends at 3, 4, 5 you have to attempt Q2.

If your roll number ends at 6,7,8,9 you have to attempt Q3.

Q1. Write and execute a Java program that accepts scores (out of 100 points) for 10 participants in a test. The SCORES ARE ASSUMED to be ENTERED by the USER in NON-INCREASING ORDER (A score value may be repeated). All scores must be displayed. After that, the program must accept a value for a score and apply the Binary Search Algorithm to determine whether more than one participant has that score. The program must display "yes" or "No". IMPORTANT values must be displayed in each iteration of Binary Search.

Q2. Given a list, split it into two sublists — one for the front half, and one for the back half. If the number of elements is odd, the extra element should go in the front list. So FrontBackSplit() on the list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}

Q3. Implement the following graph. Ask the user source node and destination node of the path and apply Breadth First Search (BFS) to source to destination.