Reg.No: L1F20BSCS0309 Time: 150 mins

Q1: (15 Marks)

You have two arrays A of size N and B of size M with positive integer values; arrays are sorted in the increasing order [duplicate values may be there]. Your task is to combine both arrays in another array C of size N+M, but you can only add values to this new array which are common and unique in both arrays [unique in A and unique in B], in the sorted order. Your solution should be of O(N+M) complexity. Think on the paper and then write the code

[**Note**: Both array traverse Only One Time]

Example:

A 2 2 3 6 6 6 7 8 9 21 25 210 210

B 1 1 4 4 5 7 10 11 11 17 21 29

C 7 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

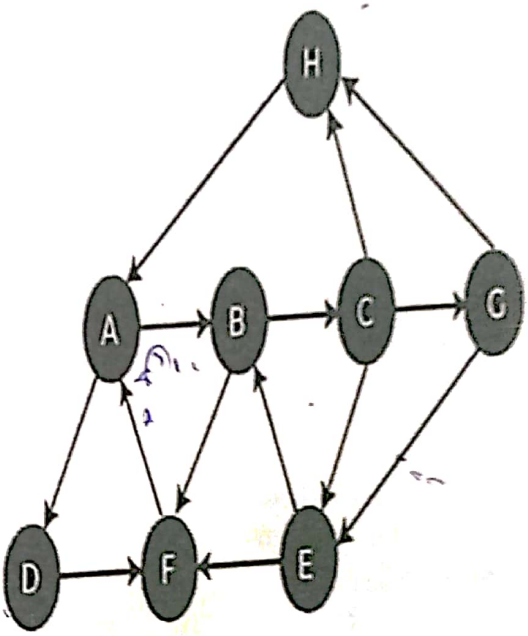
Q2: (6+7+7 Marks)

Apply DFS algorithm on the given graph and answer the following questions. The source node is 'F'. (**Note**: The adjacent nodes of a vertex are to be traversed in alphabetical descending order.)

a) Write the type of each edge of the above graph.
b) Write updated details for all the vertices when DFS-VISIT (G, u) called for the 3rd time (Just Called) write the values of their (**'Parent', 'color', 'd-time', 'f-time'**).
c) **Write down details for all the vertices when Line 10 of DFS-VISIT (G, u) completed its work for 6th time - write the values of their ('Parent', 'color', 'd-time', 'f-time').**

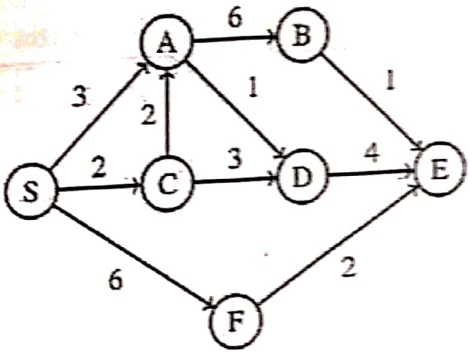
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
1  time = time + 1
2  u.d = time
3  u.color = GRAY
4  for each v ∈ G.Adj[u]
5      if v.color == WHITE
6          v.π = u
7          DFS-VISIT(G, v)
8  u.color = BLACK
9  time = time + 1
10 u.f = time
```



Q3: (9+7 Marks)

```
DIJKSTRA (G, w, s)
1.  INITIALIZE-SINGLE-SOURCE(G, s)
2.  X = Ø
3.  H = G.V
4.  While H ≠ Ø
5.      u = EXTRACT-MIN(H)
6.      X = X U {u}
7.      for each vertex v ∈ G. Adj[u]
8.          if v. d > u. d + w(u, v)
9.              v. d = u. d + w(u, v)
10.             v. π = u
```



Apply Dijkstra's Shortest Path Algorithm on the given graph and answer the following questions. The source node is **'C'**. **(Note: The adjacent nodes of a vertex are to be traversed in alphabetical order.)** – After Initialization is completed, keep track of every update in the graph

Design & Analysis of Algorithm
Final-Term (Summer 2023)

Reg.No: L1F20BSCS0309 Time: 150 mins

a) Write down the first 8 updates done on the above graph, list every update from 1st to 8th [**vertex_name (Parent, distance)**], comma separated.

b) What are the 'distance' and 'predecessor' values of nodes that are not in the heap, H, when Line-10 is executed for the 8th time. **[List all nodes with [vertex_name (Parent, distance)]]**

Q4: (6+7+7 Marks)

Try to draw or write every function call step by step so that your work can support your solution, array start from index 0, ends at index 13, keep in mind. Read the given code carefully.

```
quickSort(Arr, l, r)
1.  if l < r {
2.      let p = PARTITION(Arr, l, r)
3.      quickSort(Arr, l, p - 1)
4.      quickSort(Arr, p + 1, r)
5.  }
```

```
PARTITION(Arr,l,r)
1.      let pivot = Arr[r]
2.      let i = l - 1
3.      for j = l to r - 1 {
4.          if Arr[j] <= pivot {
5.              i = i + 1
6.              exchange Arr[i], Arr[j])
7.          }
8.      }
9.      exchange(Arr[i+1], Arr[r])
10.     return i + 1
```

Let Arr = 8 12 5 19 7 13 9 21 18 16 25 6 be used for answering the following parts.

a) Give the exact updated position of ***Arr*** when Line 2 of ***quickSort(...)*** has completed its working for third time.

b) What are exact updated contents of ***Arr*** when Line 4 of ***quickSort(...)*** is reached for the third time, also what will the value of l & r also. [Please mention every function call, 1st, 2nd, 3rd]

c) What are the first four values of ***p* (index)**, that you will get as return from the execution of the **PARTITION(Arr, l, r)** function

How you should write your function calls, **two examples are given below**

quickSort(Arr, 0, 10) & PARTITION(Arr, 0, 10)