Reg. No: _____ Time: 150 mins

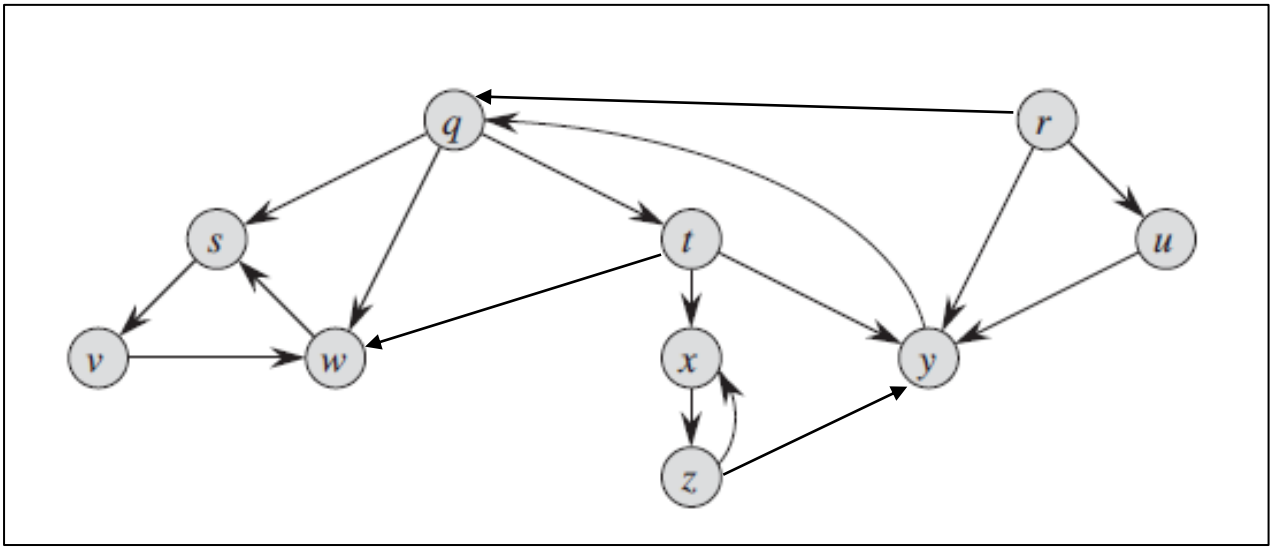
Q1: (3+6+6 Marks)

```
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
```

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

1. Write the type of each edge of the below graph, complete work should be there.
2. Write updated details for all the vertices when DFS-VISIT (G, u) [**7th line inside the DFS-VISIT**] called for the **6th time** (Just Called) write the values for all vertices (**'Parent'**, **'color'**, '**d-time', 'f-time')**.
3. Write down details for all the vertices when **Line 8** of DFS-VISIT (G, u) completed its work for **5th time** - write the values of their (**'Parent'**, '**color'**, '**d-time', 'f-time')**.



Reg. No: _____ Time: 150 mins

Q2: (20 Marks)

Write a proper function for the following task. You have two arrays **A (size N) and B (size M).** Array A is being populated with random integer numbers [no limit] & array B is also being populated with random integer numbers [0 to 9]. You have to go through all integer numbers in array A and if that number is also there in array B, delete all instances of that number in array B. [Place any other value other than the given range], if you have search/delete a number in array B already, you should not search for that number again in B [in case number is again there in A].

You can use some extra constant size space if needed for the task.

Your solution should be the best optimal for time complexity. [Think and decide]

Input: A = {11, 6, 21, 4, -2, 12, 5, 7, 6, 21, 5, 6, 16, 7, 12, 88, -12, 0, 32, 45, 54}
B = {6, 2, 4, 7, 2, 5, 7, 6, 1, 8, 8, 6, 9, 9, 8}
Output: **A = {**11, 6, 21, 4, -2, 12, 5, 7, 6, 21, 5, 6, 16, 7, 12, 88, -12, 0, 32, 45, 54}
B = {-1, 2, -1, -1, 2, -1, -1, -1, 1, 8, 8, -1, 9, 9, 8}

Q3: (6+5+4 Marks)

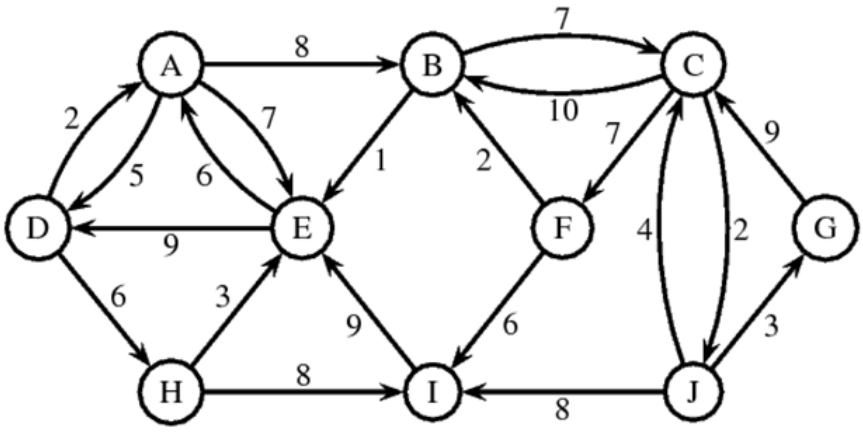
```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0

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 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

1. Write down the first 9 updates (start the count after initialization), list every update from first to the 9th [**vertex_name (Parent, distance)**], comma separated.
2. When **line 8** has been executed for **12th times**, please show all the vertices with [**vertex_name (Parent, distance)**].
3. Please find out the **two most updated vertices** during this process, mention vertex names, and all their updates from start to end. [if two vertices has the same count for change, then select the vertex alphabetically]

Graph 2



Q4: (6+9 Marks)

Keep in mind how Prim's algorithm works, you have to design a graph with **5 vertices (A, B, C, D, E) and 10 edges (every edge weight should be unique)**. When you complete your algorithm work on this graph, you should have 1 vertex updated 4 times, 1 vertex updated 3 times, 1 vertex updated 2 times, and 1 vertex updated 1 time. **(Please neither consider starting vertex nor initialization work in this problem as updates)**. You have to show all the work on your designed graph. **Start from A** [1 – Design such graph, 2 – Show Prim's work on this graph]

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Q5: (5+8+7 Marks)

a) Please recall **Knapsack (0/1)** in your mind and write its complete recurrence with all the base cases and details. [In variable terms]

You have been given data to solve some questions, **capacity for the Knapsack is 18**. Total 8 items are there for the selection

W	**4**	**7**	**5**	**3**	**2**	**6**	**5**	**4**
V	3	10	8	7	8	3	6	8

b) You have to calculate value for **V[3, 10], through recursive function calls.** First try to understand what is asked and then show step by step all the calls what are to be used in the form of recursion tree. [At every step it should be clearly mentioned that what was the decision at the step, with reason]

c) You have to calculate value for **V[4,13], through table filling method.** First try to understand what is asked, then fill the required cells in the table. At the end show which items were selected step by step, and how it was decided.