Instructions:

Write your own C/C++ Program code

Q1. Write a C/C++ program to implement "**MAKE-SET**", "**FIND-SET**", and "**UNION**" operations for **Disjoint-Set Data Structure** using (i) **Union-by-size**, (ii) **Union-by-rank** with **Path compression**. With this program, you may also find the number of connected components for a disconnected graph. The manu of the **main()** is as follows:

```
printf("How many disjoint set you want to create?");
scanf("%d",&n);
makeset(n);
printf("%d number of makeset operations are executed");
printf("To stop Union operation, press -1");
while(t!=-1)
  {
    printf("Enter the value of t");
    scanf("%d", &t);
    if(t!=-1)
      {
        printf("Enter i and j to perform Union(i,j) operation");
        printf("I = ");
        scanf("%d", &i);
        printf("J = ");
        scanf("%d", &j);
        munion(i,j);
        printf("UNION(%d,%d) is finished", i,j);
      }
  }
printf("Do you want to find the connected components");
printf("Press '1' for YES or, '0' for NO");
        scanf("%d", &flag);
    if(flag==1)
      {
         for(i=1;i<=n;i++)
```

```
                {
                   if(findset(i)==i)
                      count=count+1;
                 }
               printf("The number of connected component is %d", count);
           }
         else
           {
               printf("We do not want to find the connected components");
           }
      return 0;
        }
```

(a) Write a function **void makeset(int v)** to create a singleton disjoint sets $S_v = \{v\}$. You may execute this function for n times to create n disjoint sets $S_1 = \{1\}$, $S_2 = \{2\}$, $\cdots$, $S_n = \{n\}$, here `parent[`i`]=`i (**parent of element** i **is itself**).

(b) Write a function **int findset(int v)** to find the `parent` of the element v.

(c) Write a function **int munion(int u, int v)** to find the union of two disjoint sets represented by u and v.