ADA Lab test -1

Sushmitha Y.V
1BM19CS165

4)

(a)
```
# include <stdio.h>
# include <conio.h>
int a[20][20], q[20], visited [20], n, i, j f=0, r=-1;
void bfs(int v)
{
 for (i=1;i<n;i++)
  if (a[v][i] && !visited[i])
  q[++r]= i;
   if(f<=r)
  { visited[q[f]]=1;
   bfs (q[f++]);
  }
 }
 void main()
 {
  int v;
  printf("\n Enter the number of vertices:");
  scanf("%d", &n);
  for(i=1, <n;i++)
  {
   ~~q[i]=0;~~
   reach [i]=0;
visited[i]=0;
}
 printf("\ Enter graph data in matrix form:\n");
 for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
  scanf("%d",&a[i][j]);
  printf("\n Enter the starting vertex:");
  scanf("%d",&v);
   bfs(v);
   printf("\n The node which are reachable are:\n");
   for(i=1;i<n;i++)
    if(visited[i])
     printf("%d\t",i);
      getch();
    }
```

Sushmitha Y.V
1BM19CS165

4) (b) check whether a given graph is connected or not using DFS method.

```
# include <stdio.h>
# include <conio.h>
int a[20][20], reach[20], n;
void dfs(int v)
{ int i;
  reach[v]=1;
  for (i=1; i<=n; i++)
   if (a[v][i] && !reach[i])
  { printf("\n %d->%d", v,i);
    dfs(i);
  }
}
void main()
{ int i, j count=0;
  printf("\n Enter number of vertices:");
  scanf("%d", &n);
  for (i=1;<=n; i++)
  { reach[i]=0;
```

Sushmitha Y V
1BM19CS165

```
for(j=1,j<  n;j++)
 a[i][j] 0;
}
 printf("\n Enter the adjacency matrix:\n");
 for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 scanf("%d",&a[i][j]);
  dfs(1);
  print("\n");
   for(i=1;i<=n;i++)
   {
    if(reach[i])
    count++
   }
   if(count==n)
   printf("\n Graph is connected");
    else
    printf("\n Graph is not connected");
     getch();
   }
```

Sushmitha Y.V
1BM19CS165

Modification

```
# include <stdio.h>
# include <math.h>
# include <stdlib.h>
# include <string.h>
# include <string.h>
# include <time.h>
 int q[100];
 int visited[100];
  int adj[20][20];
  int n;

  int front=-1, rear=-1;
    void enqueue (int v)

   { if (front ==-1 && rear ==-1)
      { front = rear = 0;
      }
     } if (rear ==n-1)
    { printf ("Queue Full\n");
      return;
     }
      q[rear]=v;
      rear++;
```

```
} int dequeue()
{ int val;
  if (front == -1 || front rear)
  {
    printf("Queue Underflow\n");
    return -1;
  }
  val = q[front];
  if (front == rear || front > rear)
  { front = -1;
    rear = -1;
  } front ++;
    return val;
}
void bfs(int v)
{ for(int i = 0; i < n; i++)
    if (adj[v][i] == 1 && visited[i] == 0)
      enqueue(i);
      printf("%d\t", i);
      visited[i]=1;
  } }
}
```

```
int val = dequeue();
 if (val!=-1)
 {
   bfs(val);
 }
  else
 {printf("\n");
    return;
 }
}
}  int main()

 int flag=0;
  int i=2;
  int v, count=1;
  printf("Enter the number of the vertex\n");
   scanf("%d",&n);
  printf("Enter the Entries of the Adjacent Matrix\n");
       for(int j=0;j<n;j++)
       scanf("%d",&adj[i][j]);
    }
 }  printf("Enter the starting vertex\n");
     scanf("%d",&v);
     printf("BREADTH ORDER TRAVERSAL FOR FOREST\n");
         printf("%d\t",v);
           visited[v]=1;
           bfs(v);
```

```
for (int i=0; i<n;i++)
 {
   if (visited[i]!=1)
  {
    printf("\n TRAVERSAL \n");
    printf("\n%d \t",i);

     visited [i]=1;
      bfs (i);
      count ++;

      flag =1;
   }
 }
  if(flag==0)

 { printf("\n GRAPH is CONNECTED \n");
 }
  if (flag==1)

 { printf("(\n GRAPH is NOT CONNECTED
              AND HAS %d PARTS \n", count);

   }
   }
```