DATA STRUCTURE LAB

CHRISTY RAJ

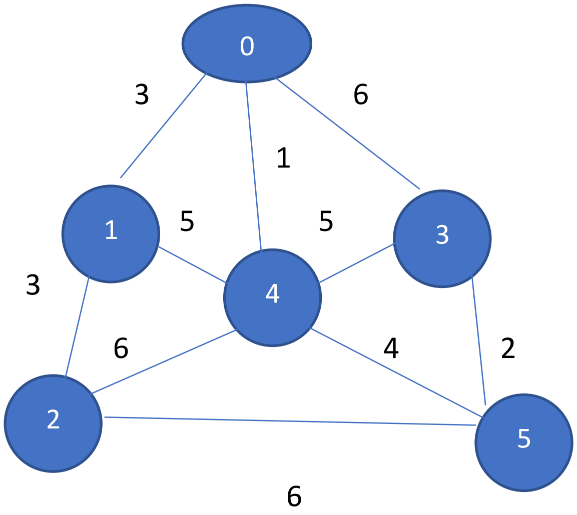
ROLL NO: MCA216

REGISTERNO: TKM20MCA-2016

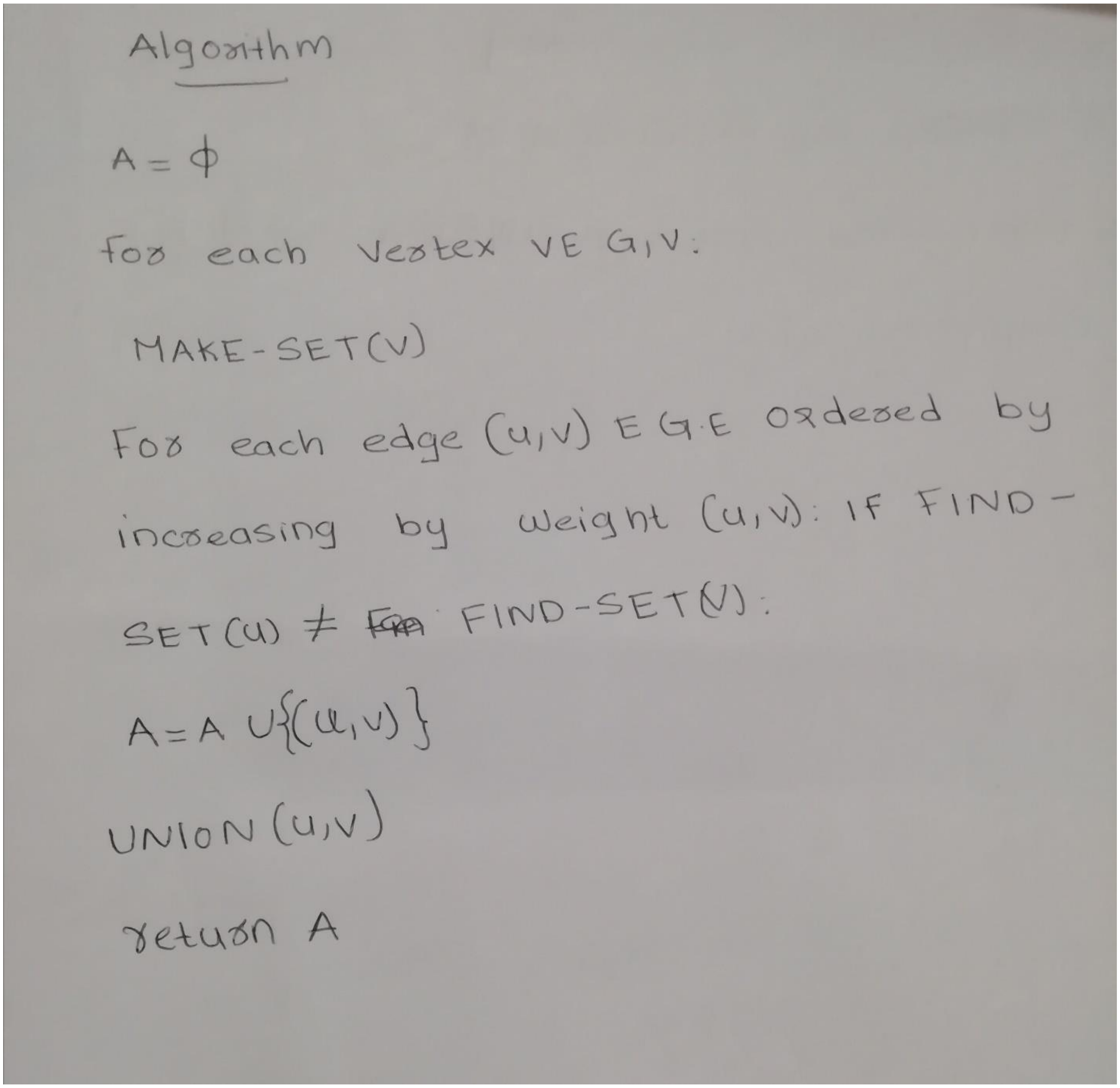
GITHUB LINK: https://github.com/christy951/Data-Structure-Lab.git

QUESTION:1

Develop a program to generate a minimum spanning tree using Kruskal algorithm for the given and compute the total cost.



ALGORITHM:



Algorithm

$A = \phi$

For each vertex V E G,V:

MAKE-SET(V)

For each edge (u,v) E G.E ordered by increasing by weight (u,v): IF FIND-SET(u) ≠ FIND-SET(V):

$A = A \cup \{(u,v)\}$

UNION (u,v)

return A

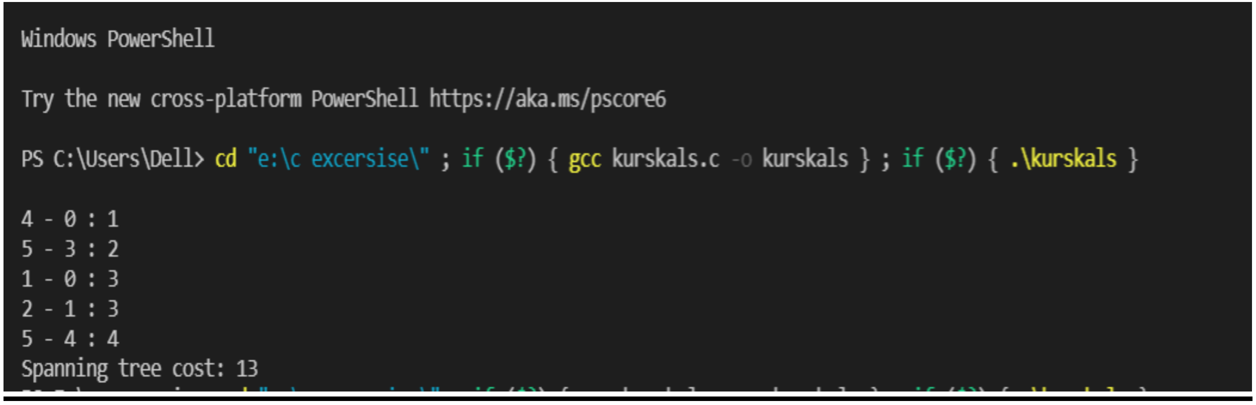
PROGRAME CODE:

```
#include <stdio.h>
#define MAX 30
typedef struct edge {
 int u, v, w;
} edge;
typedef struct edge_list {
 edge data[MAX];
 int n;
} edge_list;
edge_list elist;
int Graph[MAX][MAX], n;
edge_list spanlist;
void kruskalAlgo();
int find(int belongs[], int vertexno);
void applyUnion(int belongs[], int c1, int c2);
void sort();
void print();
// Applying Krushkal Algo
void kruskalAlgo() {
 int belongs[MAX], i, j, cno1, cno2;
 elist.n = 0;
 for (i = 1; i < n; i++)
 for (j = 0; j < i; j++) {
 if (Graph[i][j] != 0) {
 elist.data[elist.n].u = i;
 elist.data[elist.n].v = j;
 elist.data[elist.n].w = Graph[i][j];
 elist.n++;
 }
 }
 sort();
 for (i = 0; i < n; i++)
 belongs[i] = i;
 spanlist.n = 0;
 for (i = 0; i < elist.n; i++) {
 cno1 = find(belongs, elist.data[i].u);
 cno2 = find(belongs, elist.data[i].v);
 if (cno1 != cno2) {
 spanlist.data[spanlist.n] = elist.data[i];
 spanlist.n = spanlist.n + 1;
 applyUnion(belongs, cno1, cno2);
 }
 }
}
int find(int belongs[], int vertexno) {
 return (belongs[vertexno]);
}
void applyUnion(int belongs[], int c1, int c2) {
 int i;
 for (i = 0; i < n; i++)
 if (belongs[i] == c2)
 belongs[i] = c1;
}
// Sorting algo
void sort() {
 int i, j;
 edge temp;
 for (i = 1; i < elist.n; i++)
 for (j = 0; j < elist.n - 1; j++)
 if (elist.data[j].w > elist.data[j + 1].w) {
 temp = elist.data[j];
 elist.data[j] = elist.data[j + 1];
 elist.data[j + 1] = temp;
 }
}
// Printing the result
void print() {
 int i, cost = 0;
 for (i = 0; i < spanlist.n; i++) {
 printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
 cost = cost + spanlist.data[i].w;
 }
 printf("\nSpanning tree cost: %d", cost);
}
```

```
int main() {
 int i, j, total_cost;
 n = 6;
 Graph[0][0] = 0;
 Graph[0][1] = 3;
 Graph[0][2] = 0;
 Graph[0][3] = 6;
 Graph[0][5] = 0;
 Graph[0][9] = 1;
 Graph[1][0] = 3;
 Graph[1][1] = 0;
 Graph[1][2] = 3;
 Graph[1][3] = 0;
 Graph[1][4] = 0;
 Graph[1][5] = 0;
 Graph[2][0] = 0;
 Graph[2][1] = 3;
 Graph[2][2] = 0;
 Graph[2][3] = 0;
 Graph[2][4] = 6;
 Graph[2][5] = 6;
 Graph[3][0] = 6;
 Graph[3][1] = 0;
 Graph[3][2] = 0;
 Graph[3][3] = 0;
 Graph[3][4] = 5;
 Graph[3][5] = 2;
 Graph[4][0] = 1;
 Graph[4][1] = 5;
 Graph[4][2] = 6;
 Graph[4][3] = 5;
 Graph[4][4] = 0;
 Graph[4][5] = 4;
 Graph[5][0] = 0;
 Graph[5][1] = 0;
 Graph[5][2] = 6;
 Graph[5][3] = 2;
 Graph[5][4] = 4;
```

```
    Graph[5][5] = 0;
    kruskalAlgo();
    print();
}
```

OUTPUT:

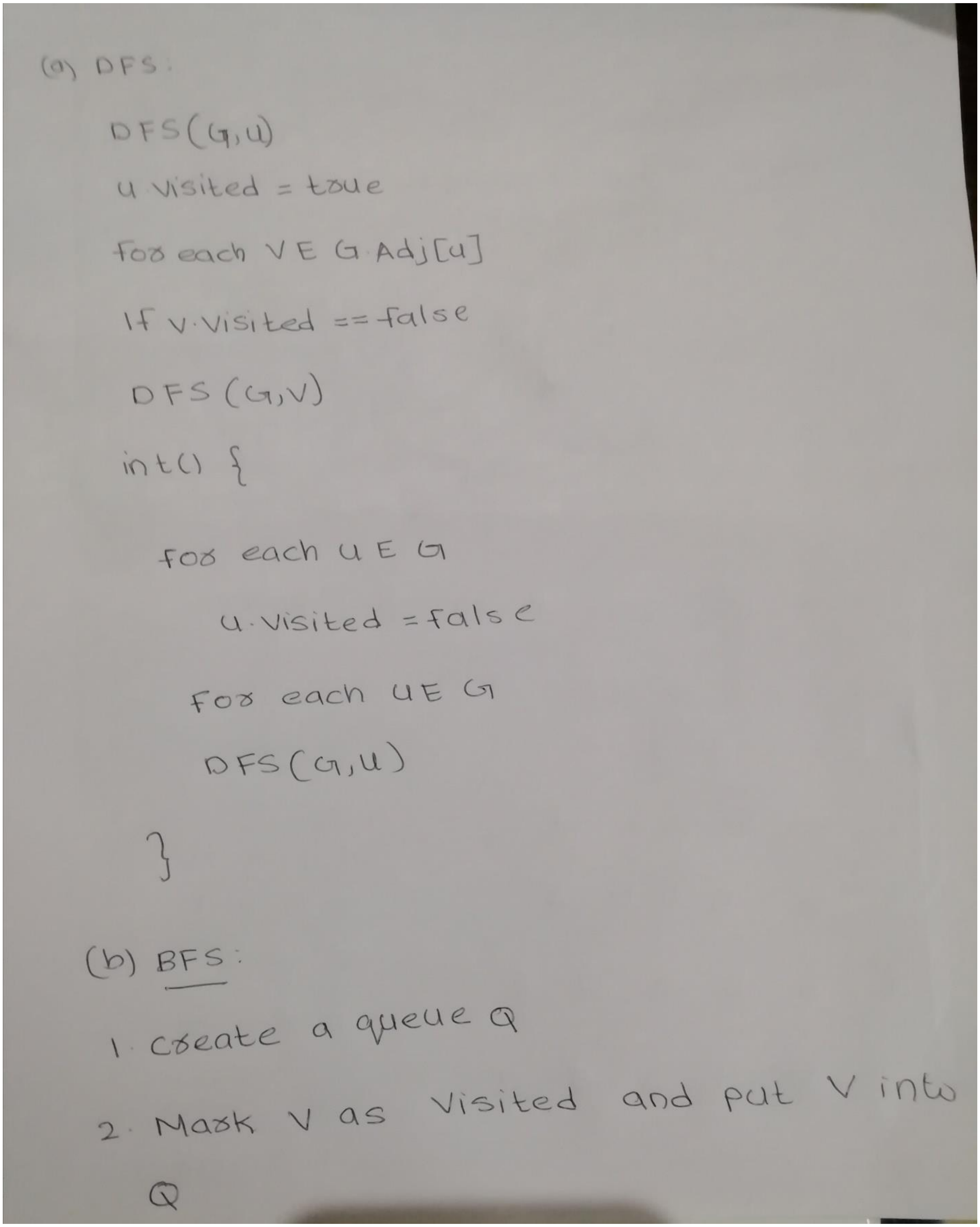


```
Windows PowerShell

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Dell> cd "e:\c excersise\" ; if ($?) { gcc kurskals.c -o kurskals } ; if ($?) { .\kurskals }

4 - 0 : 1
5 - 3 : 2
1 - 0 : 3
2 - 1 : 3
5 - 4 : 4
Spanning tree cost: 13
```

QUESTION:2

Develop a program to implement DFS and BFS.

ALGORITHM



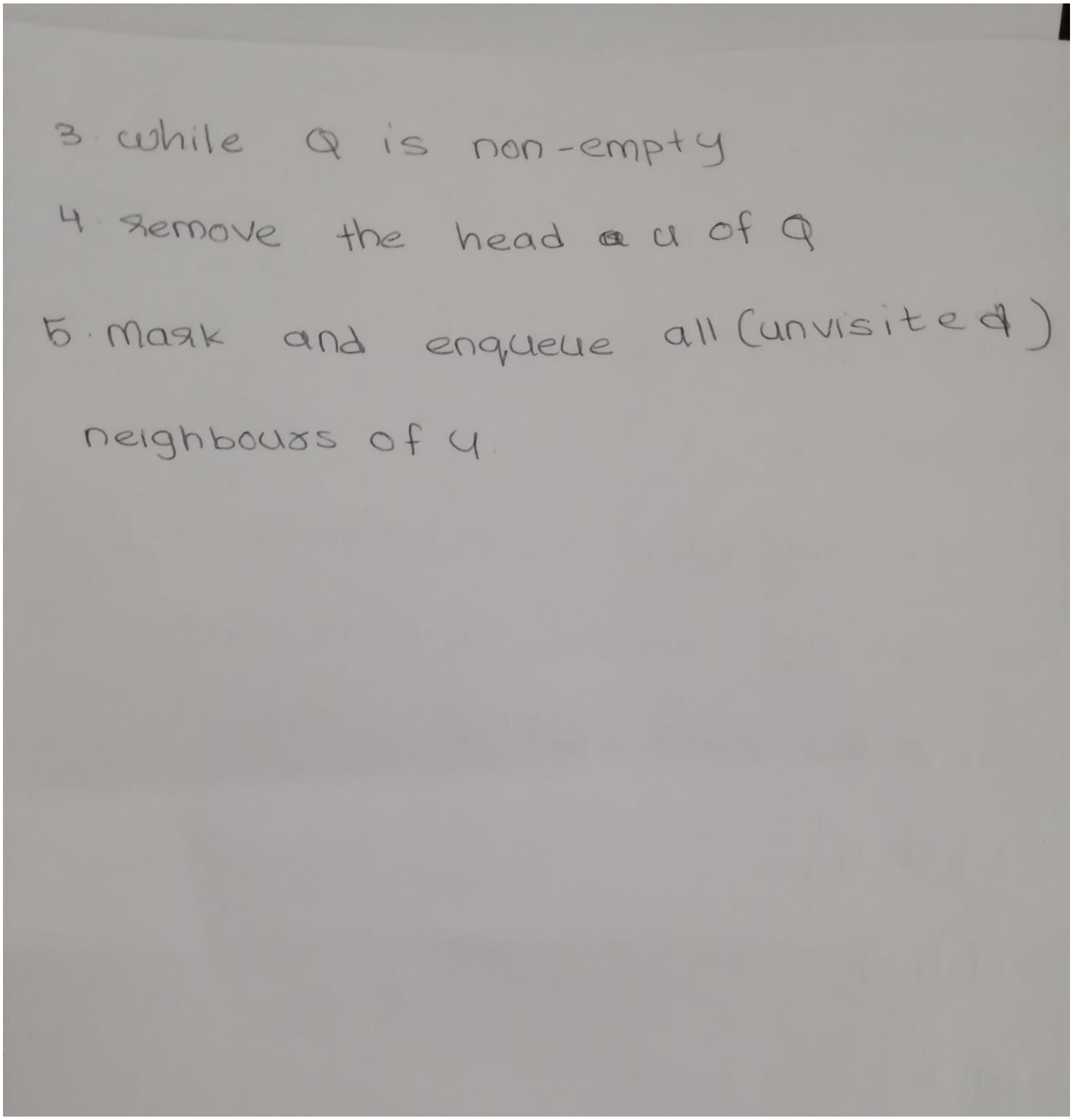
(a) DFS:

```
DFS(G,u)
u.visited = true
for each V ∈ G.Adj[u]
If v.visited == false
 DFS(G,V)
int() {
   for each u ∈ G
       u.visited = false
    For each u∈ G
    DFS(G,u)
  }
```

(b) BFS:

1. create a queue Q

2. Mark V as visited and put V into Q



3. while Q is non-empty

4. remove the head u of Q

5. Mark and enqueue all (unvisited)
neighbours of u.

PROGRAME CODE:

(a)DFS

```
#include<stdio.h>

void DFS(int);
int G[10][10],visited[10],n;    //n is no of vertices and graph is sorted in array G[10][10]

int main()
{
   int i,j;
   printf("Enter number of vertices:");

   scanf("%d",&n);

   //read the adjecency matrix
   printf("\nEnter adjecency matrix of the graph:");

   for(i=0;i<n;i++)
     for(j=0;j<n;j++)
                  scanf("%d",&G[i][j]);

   //visited is initialized to zero
  for(i=0;i<n;i++)
      visited[i]=0;

   DFS(0);
}

void DFS(int i)
{
   int j;
   printf("\n%d",i);
   visited[i]=1;

   for(j=0;j<n;j++)
```

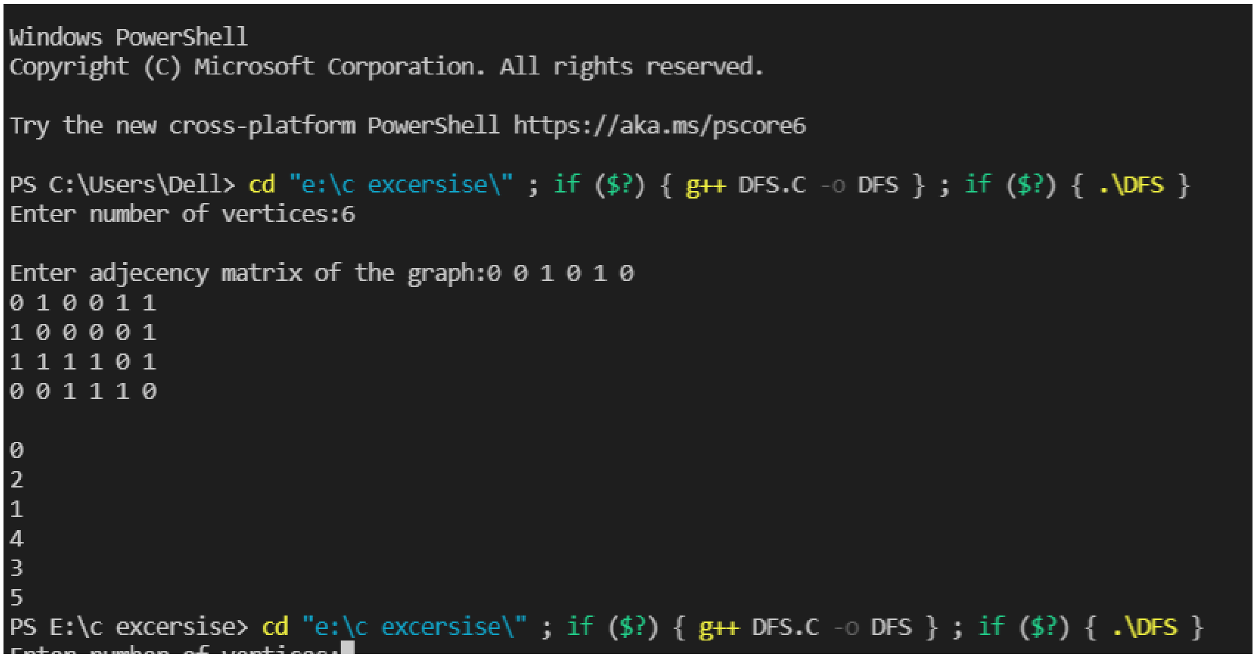
```
            if(!visited[j]&&G[i][j]==1)
                DFS(j);
    }
```

b)BFS

```
#include<stdio.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; i <= n; i++) {
 q[i] = 0;
 visited[i] = 0;
 }
printf("\n 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);
else {
printf("\n Bfs is not possible. Not all nodes are reachable");
break;
}
}
  }
```

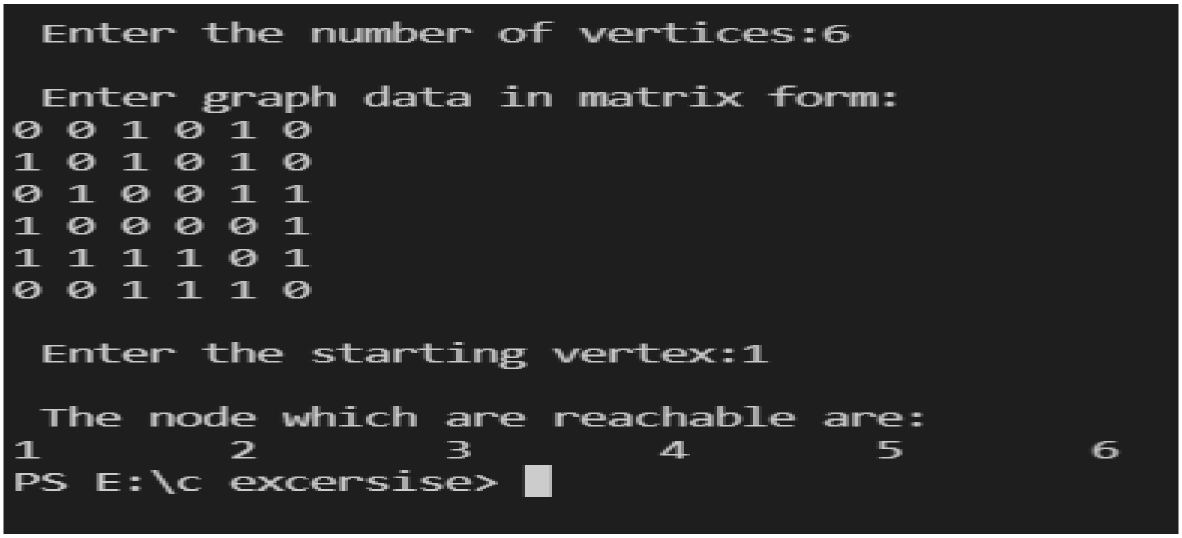
OUTPUT:

(a)DFS



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Dell> cd "e:\c excersise\" ; if ($?) { g++ DFS.C -o DFS } ; if ($?) { .\DFS }
Enter number of vertices:6

Enter adjecency matrix of the graph:0 0 1 0 1 0
0 1 0 0 1 1
1 0 0 0 0 1
1 1 1 1 0 1
0 0 1 1 1 0

0
2
1
4
3
5
PS E:\c excersise> cd "e:\c excersise\" ; if ($?) { g++ DFS.C -o DFS } ; if ($?) { .\DFS }
Enter number of vertices:
```

b)BFS



```
 Enter the number of vertices:6

 Enter graph data in matrix form:
0 0 1 0 1 0
1 0 1 0 1 0
0 1 0 0 1 1
1 0 0 0 0 1
1 1 1 1 0 1
0 0 1 1 1 0

 Enter the starting vertex:1

 The node which are reachable are:
1       2       3       4       5       6
PS E:\c excersise> 
```