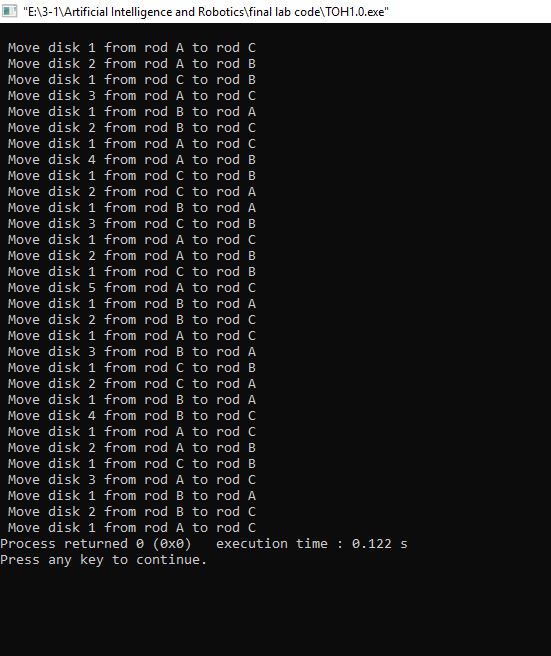
1. Experiment name: Tower of hanoi

Code:

1  
2 #include <stdio.h>  
3  
4 // C recursive function to solve tower of hanoi puzzle  
5 **void** towerOfHanoi(**int** n, **char** from\_rod, **char** to\_rod, **char** aux\_rod)  
6 {  
7 **if** (n == 1)  
8 {  
9 printf("\n Move disk 1 from rod %c to rod %c", from\_rod, to\_rod);  
10 **return**;  
11 }  
12 towerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod);  
13 printf("\n Move disk %d from rod %c to rod %c", n, from\_rod, to\_rod);  
14 towerOfHanoi(n-1, aux\_rod, to\_rod, from\_rod);  
15 }  
16  
17 **int** main()  
18 {  
19 **int** n = 4; // Number of disks  
20 towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of rods  
21 **return** 0;  
22 }

Input and output:

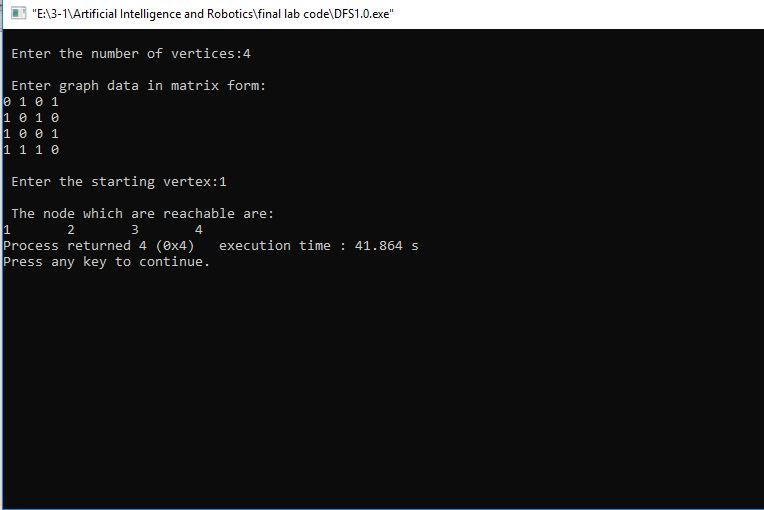


2. Experiment name: Breath first search

Code:

1 #include<stdio.h>  
2 **int** a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;  
3  
4 **void** bfs(**int** v) {  
5 **for**(i = 1; i <= n; i++)  
6 **if**(a[v][i] && !visited[i])  
7 q[++r] = i;  
8 **if**(f <= r) {  
9 visited[q[f]] = 1;  
10 bfs(q[f++]);  
11 }  
12 }  
13  
14 **void** main() {  
15 **int** v;  
16 printf("\n Enter the number of vertices:");  
17 scanf("%d", &n);  
18  
19 **for**(i=1; i <= n; i++) {  
20 q[i] = 0;  
21 visited[i] = 0;  
22 }  
23  
24 printf("\n Enter graph data in matrix form:\n");  
25 **for**(i=1; i<=n; i++) {  
26 **for**(j=1;j<=n;j++) {  
27 scanf("%d", &a[i][j]);  
28 }  
29 }  
30  
31 printf("\n Enter the starting vertex:");  
32 scanf("%d", &v);  
33 bfs(v);  
34 printf("\n The node which are reachable are:\n");  
35  
36 **for**(i=1; i <= n; i++) {  
37 **if**(visited[i])  
38 printf("%d\t", i);  
39 **else** {  
40 printf("\n Bfs is not possible. Not all nodes are reachable");  
41 **break**;  
42 }  
43 }  
44 }

Input and output:

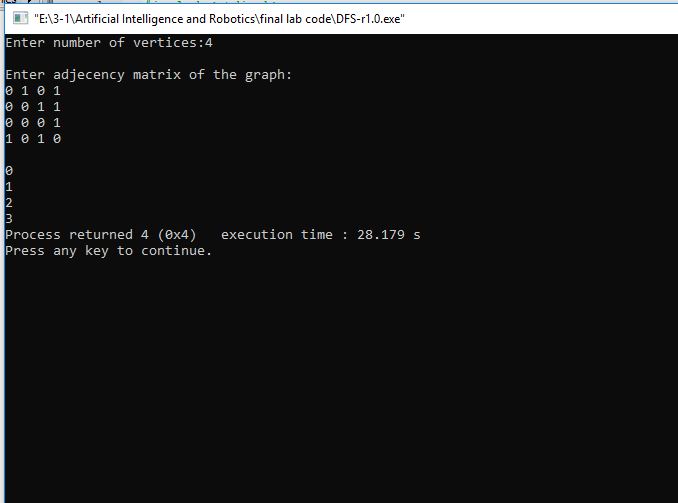


3. Experiment name: Depth first search

Code:

1 #include<stdio.h>  
2  
3 **void** DFS(**int**);  
4 **int** G[10][10],visited[10],n;  
5  
6 **void** main()  
7 {  
8 **int** i,j;  
9 printf("Enter number of vertices:");  
10  
11 scanf("%d",&n);  
12  
13  
14 printf("\nEnter adjecency matrix of the graph:\n");  
15  
16 **for**(i=0;i<n;i++)  
17 **for**(j=0;j<n;j++)  
18 scanf("%d",&G[i][j]);  
19  
20  
21 **for**(i=0;i<n;i++)  
22 visited[i]=0;  
23  
24 DFS(0);  
25 }  
26  
27 **void** DFS(**int** i)  
28 {  
29 **int** j;  
30 printf("\n%d",i);  
31 visited[i]=1;  
32  
33 **for**(j=0;j<n;j++)  
34 **if**(!visited[j]&&G[i][j]==1)  
35 DFS(j);  
36 }

Input and output:



4. Experiment name: Sells man travelling

Code:

1 #include<stdio.h>  
3 **int** ary[10][10],completed[10],n,cost=0;  
5 **void** takeInput()  
6 {  
7 **int** i,j;  
8  
9 printf("Enter the number of node: ");  
10 scanf("%d",&n);  
11  
12 printf("\nEnter the Cost Matrix\n");  
13  
14 **for**(i=0;i < n;i++)  
15 {  
16 **for**( j=0;j < n;j++)  
17 scanf("%d",&ary[i][j]);  
18  
19 completed[i]=0;  
20 }  
21  
22 }  
23  
24 **void** mincost(**int** city)  
25 {  
26 **int** i,ncity;  
27  
28 completed[city]=1;  
29  
30 printf("%d--->",city+1);  
31 ncity=least(city);  
32  
33 **if**(ncity==999)  
34 {  
35 ncity=0;  
36 printf("%d",ncity+1);  
37 cost+=ary[city][ncity];  
38  
39 **return**;  
40 }  
41  
42 mincost(ncity);  
43 }  
44  
45 **int** least(**int** c)  
46 {  
47 **int** i,nc=999;  
48 **int** min=999,kmin;  
49  
50 **for**(i=0;i < n;i++)  
51 {  
52 **if**((ary[c][i]!=0)&&(completed[i]==0))  
53 **if**(ary[c][i]+ary[i][c] < min)  
54 {  
55 min=ary[i][0]+ary[c][i];  
56 kmin=ary[c][i];  
57 nc=i;  
58 }  
59 }  
60  
61 **if**(min!=999)  
62 cost+=kmin;  
63  
64 **return** nc;  
65 }  
66  
67 **int** main()  
68 {  
69 takeInput();  
70  
71 printf("\n\nThe Path is:\n");  
72 mincost(0); //passing 0 because starting vertex  
73  
74 printf("\n\nMinimum cost is %d\n ",cost);  
75  
76 **return** 0;  
77 }

Input and output:

