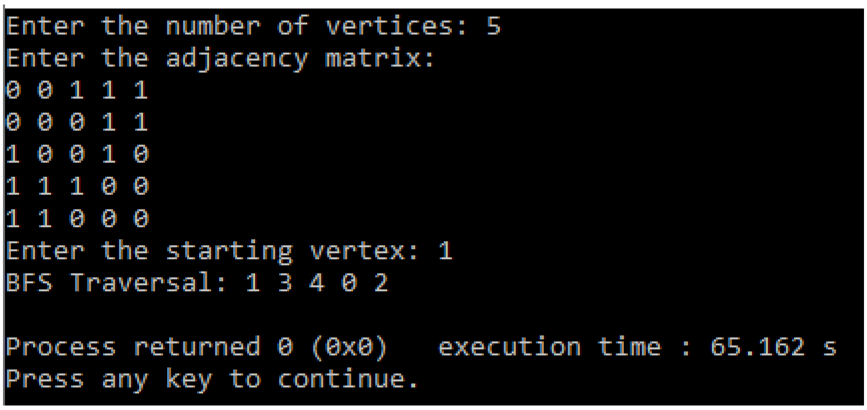
Lab program 9:

Write a program to traverse a graph using Breadth First Search (BFS) method.

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

// Queue structure for BFS
int queue[MAX], front = -1, rear = -1;

// Function to enqueue an element
void enqueue(int item) {
    if (rear == MAX - 1) {
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1) front = 0;
    queue[++rear] = item;
}

// Function to dequeue an element
int dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue Underflow\n");
        return -1;
    }
    return queue[front++];
}

// BFS Function
void bfs(int graph[MAX][MAX], int visited[MAX], int start, int n) {
    int i;
    enqueue(start);
    visited[start] = 1;

    printf("BFS Traversal: ");
    while (front <= rear) {
        int current = dequeue();
        printf("%d ", current);

        for (i = 0; i < n; i++) {
            if (graph[current][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}

// Main Function
int main() {
    int n, i, j, start;
    int graph[MAX][MAX], visited[MAX] = {0};

    printf("Enter the number of vertices: ");
    scanf("%d", &n);

    printf("Enter the adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

    printf("Enter the starting vertex: ");
    scanf("%d", &start);

    bfs(graph, visited, start, n);

    return 0;
}
```

Output:



```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 0 1 1 1
0 0 0 1 1
1 0 0 1 0
1 1 1 0 0
1 1 0 0 0
Enter the starting vertex: 1
BFS Traversal: 1 3 4 0 2

Process returned 0 (0x0)   execution time : 65.162 s
Press any key to continue.
```

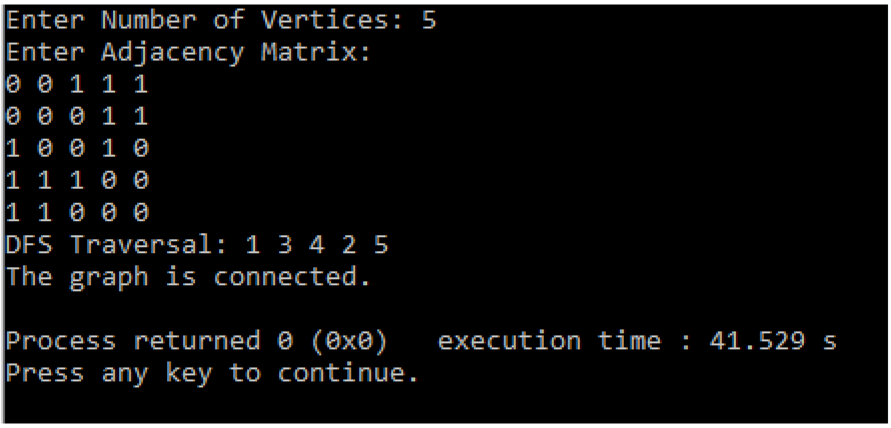
Lab program 9:

Write a program to check whether given graph is connected or not using DFS method.

```
#include <stdio.h>
#define MAX 10

int a[MAX][MAX], vis[MAX], n;

void dfsConnected(int v) {
   vis[v] = 1;


   for (int i = 0; i < n; i++) {
      if (a[v][i] == 1 && !vis[i]) {
         dfsConnected(i);
      }
   }
}

int isConnected() {

   for (int i = 0; i < n; i++) {
      vis[i] = 0;
   }


   dfsConnected(0);
   for (int i = 0; i < n; i++) {
      if (!vis[i]) {
         return 0;
      }
   }
   return 1;
}

void dfs(int v) {
   printf("%d ", v+1);
   vis[v] = 1; // Mark the current node as visited

   for (int i = 0; i < n; i++) {
      // If there is an edge from v to i and i is not visited
      if (a[v][i] == 1 && vis[i] == 0) {
         dfs(i);
      }
```

```
    }
}



void main() {
    int i, j;

    printf("Enter Number of Vertices: ");
    scanf("%d", &n);

    printf("Enter Adjacency Matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }

    for (i = 0; i < n; i++) {
        vis[i] = 0; // Initialize visited array
    }

    printf("DFS Traversal: ");
    for (i = 0; i < n; i++) {
        if (vis[i] == 0) {
            dfs(i);
        }
    }

    printf("\n");
    if (isConnected()) {
        printf("The graph is connected.\n");
    }
    else {
        printf("The graph is not connected.\n");
    }
}
```

Output:



```
Enter Number of Vertices: 5
Enter Adjacency Matrix:
0 0 1 1 1
0 0 0 1 1
1 0 0 1 0
1 1 1 0 0
1 1 0 0 0
DFS Traversal: 1 3 4 2 5
The graph is connected.

Process returned 0 (0x0)   execution time : 41.529 s
Press any key to continue.
```