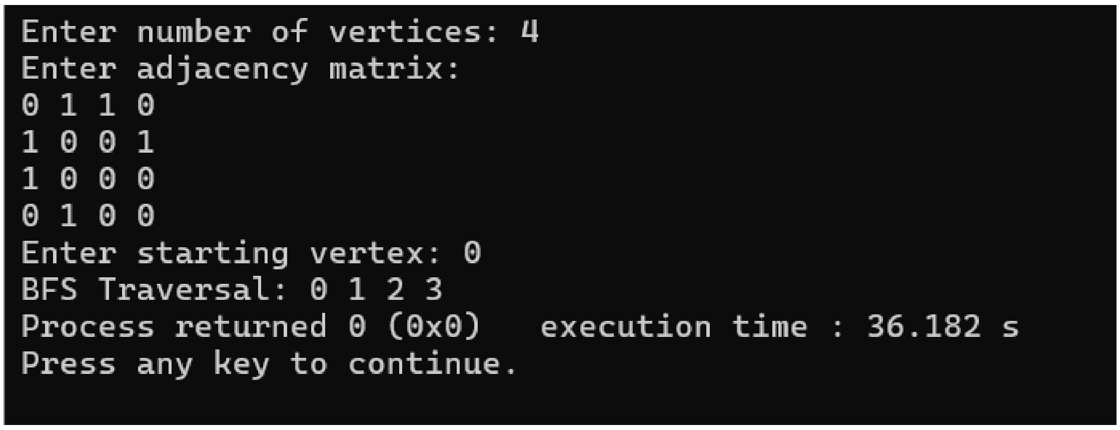
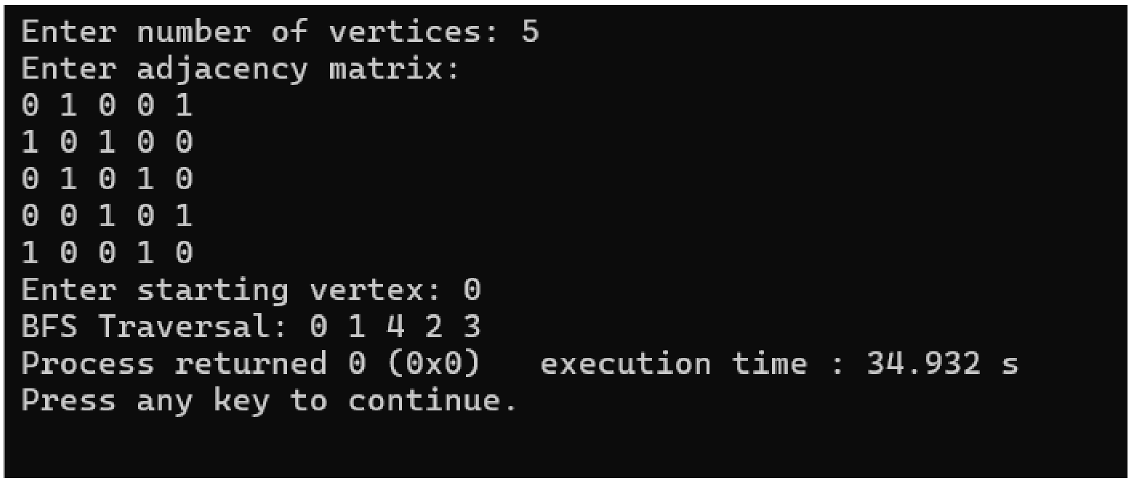
9a) Write a program to traverse a graph using BFS method.

```c
#include <stdio.h>
#define MAX 20

int queue[MAX], front = -1, rear = -1;
int visited[MAX];


void enqueue(int v) {
    if (rear == MAX - 1) {
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1)
        front = 0;
    queue[++rear] = v;
}


int dequeue() {
    if (front == -1 || front > rear)
        return -1;
    return queue[front++];
}


void BFS(int adj[MAX][MAX], int n, int start) {
    int i, j;

    for (i = 0; i < n; i++)
        visited[i] = 0;

    enqueue(start);
    visited[start] = 1;

    printf("BFS Traversal: ");

    while (front != -1 && front <= rear) {
        int node = dequeue();
        printf("%d ", node);

        for (j = 0; j < n; j++) {
            if (adj[node][j] == 1 && !visited[j]) {
                enqueue(j);
                visited[j] = 1;
            }
        }
    }
}

int main() {
    int n, start;
    int adj[MAX][MAX];

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

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

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

    BFS(adj, n, start);
    return 0;
}
```

Output:



```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 0
0 1 0 0
Enter starting vertex: 0
BFS Traversal: 0 1 2 3
Process returned 0 (0x0)   execution time : 36.182 s
Press any key to continue.
```



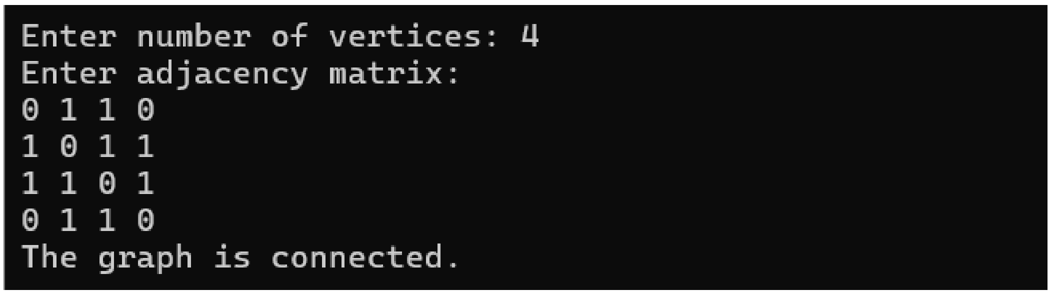
```
Enter number of vertices: 5
Enter adjacency matrix:
0 1 0 0 1
1 0 1 0 0
0 1 0 1 0
0 0 1 0 1
1 0 0 1 0
Enter starting vertex: 0
BFS Traversal: 0 1 4 2 3
Process returned 0 (0x0)   execution time : 34.932 s
Press any key to continue.
```

9(b) Write a program to check whether a given graph is connected or not using the DFS method.

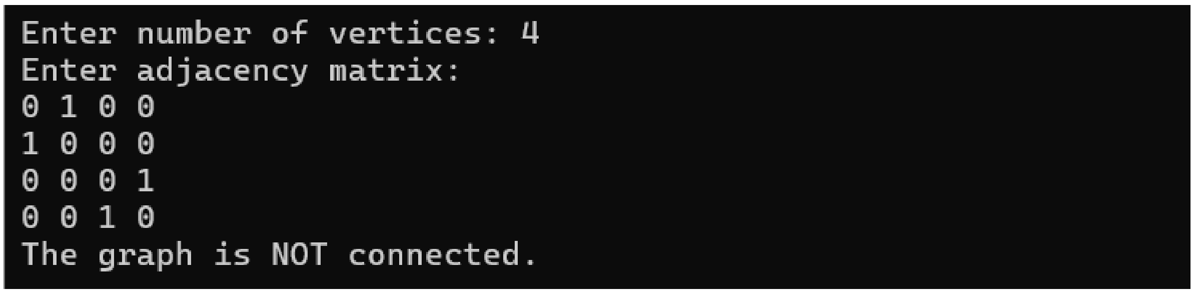
```
#include <stdio.h>
#define MAX 20

int visited[MAX];

void DFS(int adj[MAX][MAX], int n, int start) {
    visited[start] = 1;
    for (int j = 0; j < n; j++) {
        if (adj[start][j] == 1 && !visited[j]) {
            DFS(adj, n, j);
        }
    }
}

int main() {
    int n;
    int adj[MAX][MAX];

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

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

    for (int i = 0; i < n; i++)
        visited[i] = 0;

    DFS(adj, n, 0);

    int connected = 1;
    for (int i = 0; i < n; i++) {
        if (!visited[i]) {
            connected = 0;
            break;
        }
    }

    if (connected)
        printf("The graph is connected.\n");
    else
```

```
        printf("The graph is NOT connected.\n");

    return 0;
}
```

Output:



```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 1 1
1 1 0 1
0 1 1 0
The graph is connected.
```



```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 0 0
1 0 0 0
0 0 0 1
0 0 1 0
The graph is NOT connected.
```