Data Structures- Week 10:

Name: Aaryan Prakash
Class: 3A
USN: 1BM23CS006

Question 1: Write a program to traverse a graph using the BFS method.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int queue[MAX], front = -1, rear = -1;
void enqueue(int item) {
   if (rear == MAX - 1) {
      printf("Queue is Full\n");
      return;
   }
   if (front == -1){
         front = 0;
   }
   queue[++rear] = item;
}
int dequeue() {
   if (front == -1 || front > rear) {
      printf("Queue is Empty\n");
      return -1;
   }
   return queue[front++];
}
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] == 0){
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}
void 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);
}
```

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 10 (0xA)   execution time : 30.652 s
Press any key to continue.
```

Question 2: Write a program to check whether a given graph is connected or not using the DFS method

Code:

```
#include <stdio.h>
#define MAX 10
int a[MAX][MAX], vis[MAX], n;
void dfs(int v);
int isConnected();

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]);
      }
   }
   printf("\nDFS Traversal: ");
   if (isConnected()) {
      printf("\nThe graph is connected.\n");
   } else {
      printf("\nThe graph is disconnected.\n");
   }
   for (i = 0; i < n; i++) {
      vis[i] = 0;
   }
   printf("DFS Traversal: ");
   for (i = 0; i < n; i++) {
      if (vis[i] == 0) {
```

```
            dfs(i);
        }
    }
    printf("\n");
}
void dfs(int v) {
    printf("%d ", v);
    vis[v] = 1;
    for (int i = 0; i < n; i++) {
        if (a[v][i] == 1 && vis[i] == 0) {
            dfs(i);
        }
    }
}
int isConnected() {
    int i;
    for (i = 0; i < n; i++) {
        vis[i] = 0;
    }
    dfs(0);
    for (i = 0; i < n; i++) {
        if (vis[i] == 0) {
            return 0;
        }
    }
    return 1;
}
```

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: 0 2 3 1 4
The graph is connected.
```