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

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

struct Node { int data;
  struct Node* left; struct Node* right;
};

struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->left = newNode->right = NULL; return newNode;
}

void BFS(struct Node* root) { if (root == NULL)
    return;

  struct Node* queue[1000]; int front = 0, rear = 0; queue[rear++] = root;

  while (front < rear) {
    struct Node* current = queue[front++]; printf("%d ", current->data);

    if (current->left != NULL) queue[rear++] = current->left;
    if (current->right != NULL)
```

```
        queue[rear++] = current->right;
    }

    printf("\n");
}

int main() {
    struct Node* root = createNode(1); root->left = createNode(2);
    root->right = createNode(3); root->left->left = createNode(4);
    root->left->right = createNode(5); root->right->left = createNode(6);
    root->right->right = createNode(7);

    printf("Breadth First Traversal of the binary tree is: \n"); BFS(root);
```

```
Breadth First Traversal of the binary tree is:
1 2 3 4 5 6 7
```

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

```
#include <stdio.h> #include <stdbool.h>
#define MAX_VERTICES 100 struct Graph {
  int V;
  int adjMatrix[MAX_VERTICES][MAX_VERTICES];
};

void initGraph(struct Graph *G, int V) { G->V = V;
  for (int i = 0; i < V; i++) { for (int j = 0; j < V; j++) {
      G->adjMatrix[i][j] = 0;
    }
  }
}

void addEdge(struct Graph *G, int src, int dest) { G->adjMatrix[src][dest]
  = 1;
  G->adjMatrix[dest][src] = 1; // If the graph is undirected
}

void DFS(struct Graph *G, int v, bool visited[]) { visited[v] = true;
  for (int i = 0; i < G->V; i++) {
    if (G->adjMatrix[v][i] && !visited[i]) { DFS(G, i, visited);
    }
  }
}

bool isConnected(struct Graph *G) { bool visited[MAX_VERTICES] =
  {false};
  DFS(G, 0, visited); // Start DFS from vertex 0 for (int i = 0; i < G->V; i++) {
    if (!visited[i]) {
      return false; // If any vertex is not reachable, return false
    }
  }
```

```
  return true;
}



int main() {
  struct Graph G;
  int V = 5; // Number of vertices initGraph(&G, V);

  addEdge(&G, 0, 1);
  addEdge(&G, 0, 2);
  addEdge(&G, 1, 2);
  addEdge(&G, 3, 4);

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

```
The graph is not connected.
```