Breadth First Search(BFS)

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

#define MAX 100

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

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

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

void bfs(int adj[MAX][MAX], int n, int start) {
   int i, current;

   // initialize visited array
```

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

    // reset queue pointers
    front = rear = -1;

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

    printf("BFS Traversal starting from vertex %d: ", start);

    // correct condition: assign first, then compare with -1
    while ((current = dequeue()) != -1) {
        printf("%d ", current);

        // check all neighbours of 'current'
        for (i = 0; i < n; i++) {
            // correct adjacency check
            if (adj[current][i] == 1 && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}

int main() {
    int n, i, j, start;
```

```
    int adj[MAX][MAX];

    printf("Enter no. of vertices: ");
    scanf("%d", &n);

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

    printf("Enter starting vertex (0 to %d): ", n - 1);
    scanf("%d", &start);

    bfs(adj, n, start);

    return 0;
}
```

Output:

```
Enter no. of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex (0 to 3): 0
BFS Traversal starting from vertex 0: 0 1 2 3
```