EXPERIMENT-10

Program :

```
#include <stdio.h>
#define MAX 100

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

// Enqueue function
void enqueue(int vertex) {
    if (rear == MAX - 1)
        printf("Queue Overflow\n");
    else {
        if (front == -1)
            front = 0;
        queue[++rear] = vertex;
    }
}

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

// BFS function
void BFS(int adj[MAX][MAX], int n, int start) {
    int visited[MAX] = {0};
    int i, v;

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

```
    printf("BFS Traversal (from vertex %d): ", start + 1);

    while (front ⩽ rear) {
        v = dequeue();
        printf("%d ", v + 1); // Print vertex (1-indexed)

        for (i = 0; i < n; i++) {
            if (adj[v][i] == 1 && visited[i] == 0) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n"); // newline after traversal
}

// Main function
int main() {
    int n, i, j, start;
    int adj[MAX][MAX];

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

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

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

    printf("\n"); // Add a blank line for neatness
```

```
    BFS(adj, n, start);

    return 0;
}
```

Output :

```
PS C:\Users\there> cd Downloads\Alwin
PS C:\Users\there\Downloads\Alwin> gcc BFS.c
PS C:\Users\there\Downloads\Alwin> ./a.exe
Enter number of vertices: 5
Enter adjacency matrix (5 x 5):
0 1 1 0 0
1 0 0 1 0
1 0 0 1 1
0 1 1 0 1
0 0 1 1 0
Enter starting vertex (1 to 5): 2
BFS Traversal: 2 1 4 3 5
PS C:\Users\there\Downloads\Alwin>
```