

```
#include <stdio.h>

#define MAX 10

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

// Function to insert element into queue
void enqueue(int vertex) {
    if (rear == MAX - 1)
        return;
    if (front == -1)
        front = 0;
    queue[++rear] = vertex;
}

// Function to delete element from queue
int dequeue() {
    if (front == -1 || front > rear)
        return -1;
    return queue[front++];
}

// Breadth First Search Function
void BFS(int adj[MAX][MAX], int n, int start) {
    int i, vertex;
    enqueue(start);
    visited[start] = 1;
```

```

printf("BFS Traversal: ");
while ((vertex = dequeue()) != -1) {
    printf("%d ", vertex);
    for (i = 0; i < n; i++) {
        if (adj[vertex][i] == 1 && visited[i] == 0) {
            enqueue(i);
            visited[i] = 1;
        }
    }
}
}

```

```

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

    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", &adj[i][j]);

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

    for (i = 0; i < n; i++)

```

```
visited[i] = 0;
```

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

```
return 0;
```

```
}
```

The screenshot displays the Programiz Online C Compiler interface. The browser tabs at the top include '172.23.30.101', 'dharshini27/Data-Structure', 'Matrix multiplication program', and 'Online C Compiler - Programiz'. The address bar shows 'programiz.com/c-programming/online-compiler/'.

The main interface features a header with the Programiz logo, a banner for 'Fly from idea to concept with AI.' with 'Try now' and 'Adobe Firefly' buttons, and a 'Programiz PRO' button.

The editor area, titled 'main.c', contains the following C code:

```
1 #include <stdio.h>
2 #define MAX 10
3
4 int queue[MAX], front = -1, rear = -1;
5 int visited[MAX];
6
7 // Function to insert element into queue
8- void enqueue(int vertex) {
9     if (rear == MAX - 1)
10         return;
11     if (front == -1)
12         front = 0;
13     queue[++rear] = vertex;
14 }
15
16 // Function to delete element from queue
17- int dequeue() {
18     if (front == -1 || front > rear)
19         return -1;
20     return queue[front++];
21 }
22
```

The output area on the right shows the execution results:

```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 1 1
1 1 0 1
0 1 1 0
Enter starting vertex (0 to 3): 0
BFS Traversal: 0 1 2 3

=== Code Execution Successful ===
```