//Write a program for BFS

```
#include <iostream>
#include <queue>
#define NODE 6
using namespace std;
class node {
public:
   int val;
   int state; // 0: not visited, 1: visited, 2: completed
};

int graph[NODE][NODE] = {
   {0, 1, 1, 1, 0, 0},
   {1, 0, 0, 1, 1, 0},
   {1, 0, 0, 1, 0, 1},
   {1, 1, 1, 0, 1, 1},
   {0, 1, 0, 1, 0, 1},
   {0, 0, 1, 1, 1, 0}
};

void bfs(node *vert, node s) {
   node u;
   int i;
   queue<node> que;
```

```
// Initialize all nodes as not visited
for (i = 0; i < NODE; i++) {
   vert[i].state = 0;
}

vert[s.val].state = 1; // Mark the starting node as visited
que.push(s); // Insert the starting node into the queue

while (!que.empty()) {
   u = que.front(); // Get the front node
   que.pop();       // Remove it from the queue

   cout << char(u.val + 'A') << " "; // Print the node

   // Explore all adjacent nodes
   for (i = 0; i < NODE; i++) {
      if (graph[u.val][i]) { // Check if there is an edge
         if (vert[i].state == 0) { // If the node is not visited
            vert[i].state = 1;    // Mark it as visited
            que.push(vert[i]);    // Add it to the queue
         }
      }
   }
   u.state = 2; // Mark the node as completed
}
```

```
}

int main() {
    node vertices[NODE];
    node start;
    char s;

    // Initialize the vertices
    for (int i = 0; i < NODE; i++) {
        vertices[i].val = i;
    }

    s = 'B'; // Starting vertex is B
    start.val = s - 'A';

    cout << "BFS Traversal: ";
    bfs(vertices, start);
    cout << endl;

    return 0;
}
```

//OUTPUT

```
BFS Traversal: B A D E C F
```