

```
1: #include<stdio.h>
2: #include<stdlib.h>
3:
4: struct queue
5: {
6:     int size;
7:     int f;
8:     int r;
9:     int* arr;
10: };
11:
12:
13: int isEmpty(struct queue *q){
14:     if(q->r==q->f){
15:         return 1;
16:     }
17:     return 0;
18: }
19:
20: int isFull(struct queue *q){
21:     if(q->r==q->size-1){
22:         return 1;
23:     }
24:     return 0;
25: }
26:
27: void enqueue(struct queue *q, int val){
28:     if(isFull(q)){
29:         printf("This Queue is full\n");
30:     }
31:     else{
32:         q->r++;
33:         q->arr[q->r] = val;
34:         // printf("Enqued element: %d\n", val);
35:     }
36: }
37:
38: int dequeue(struct queue *q){
39:     int a = -1;
```

```

40:     if(isEmpty(q)){
41:         printf("This Queue is empty\n");
42:     }
43:     else{
44:         q->f++;
45:         a = q->arr[q->f];
46:     }
47:     return a;
48: }
49:
50: int main(){
51:     // Initializing Queue (Array Implementation)
52:     struct queue q;
53:     q.size = 400;
54:     q.f = q.r = 0;
55:     q.arr = (int*) malloc(q.size*sizeof(int));
56:
57:     // BFS Implementation
58:     int node;
59:     int i = 0;
60:     int visited[7] = {0,0,0,0,0,0,0};
61:     int a [7][7] = {
62:         {0,1,1,1,0,0,0},
63:         {1,0,1,0,0,0,0},
64:         {1,1,0,1,1,0,0},
65:         {1,0,1,0,1,0,0},
66:         {0,0,1,1,0,1,1},
67:         {0,0,0,0,1,0,0},
68:         {0,0,0,0,1,0,0}
69:     };
70:     printf("%d", i);
71:     visited[i] = 1;
72:     enqueue(&q, i); // Enqueue i for exploration
73:     while (!isEmpty(&q))
74:     {
75:         int node = dequeue(&q);
76:         for (int j = 0; j < 7; j++)
77:         {
78:             if(a[node][j] ==1 && visited[j] == 0){

```

```
79:                printf("%d", j);
80:                visited[j] = 1;
81:                enqueue(&q, j);
82:            }
83:        }
84:    }
85:    return 0;
86: }
87:
```