code

9a.
```
#include <stdio.h>
int graph[20][20], visited[20], n;
void BFS(int start)
{   int queue[20], front=0, rear=0;
    visited[start] = 1;
    queue[rear++] = start;
    while (front < rear) {
        int node = queue[front++];
        printf("%d ", node);
        for (int i = 0; i < n; i++) {
            if (graph[node][i] == 1 && !visited[i]) {
                visited[i] = 1;
                queue[rear++] = i;
            }
        }
    }
}
int main() {
    int start;
    printf("enter number of vertices: \n");
    scanf("%d", &n);
    printf("enter adjacency matrix: \n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    }
    for (int i=0; i<n; i++)
        visited[i] = 0;
    printf("enter starting vertex: \n");
    scanf("%d", &start);
    printf("BFS traversal: \n");
    BFS(start);
    return 0;
}
```

Output:

```
enter number of vertices: 4
enter adjacency matrix:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
enter starting vertex: 0
BFS traversal:
0 1 2 3
```

9b.
```
#include <stdio.h>
#define N 10
int visited[N];
int adj[N][N];
int n;
void DFS(int v){
    visited[v] = 1;
    printf("%d ", v);
    for(int i=0; i<n; i++){
        if (adj[v][i]==1 && !visited[i])
            DSF(i);
    }
}
int main(){
    int connected = 1;
    printf("enter number of vertices: \n");
    scanf("%d", &n);
    printf("enter adjacency matrix: \n");
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++)
            scanf("%d", &adj[i][j]);
    }
    for(int i=0; i<n; i++)
        visited[i]=0;
    printf("DFS traversal starting from vertex 0: \n");
    DFS(0);
    for(int i=0; i<n; i++){
        if(!visited[i]){
            connected=0;
            break;
        }
    }
    if(connected)
        printf("\n graph is connected \n");
    else
        printf("\n graph is not connected \n");
    return 0;
}
```

Output:

```
enter number of vertices: 4
enter adjacency matrix:
0 1 0 1
1 0 1 1
0 1 0 0
1 1 0 0
DFS traversal starting from vertex 0:
0 1 2 3
graph is connected.
```