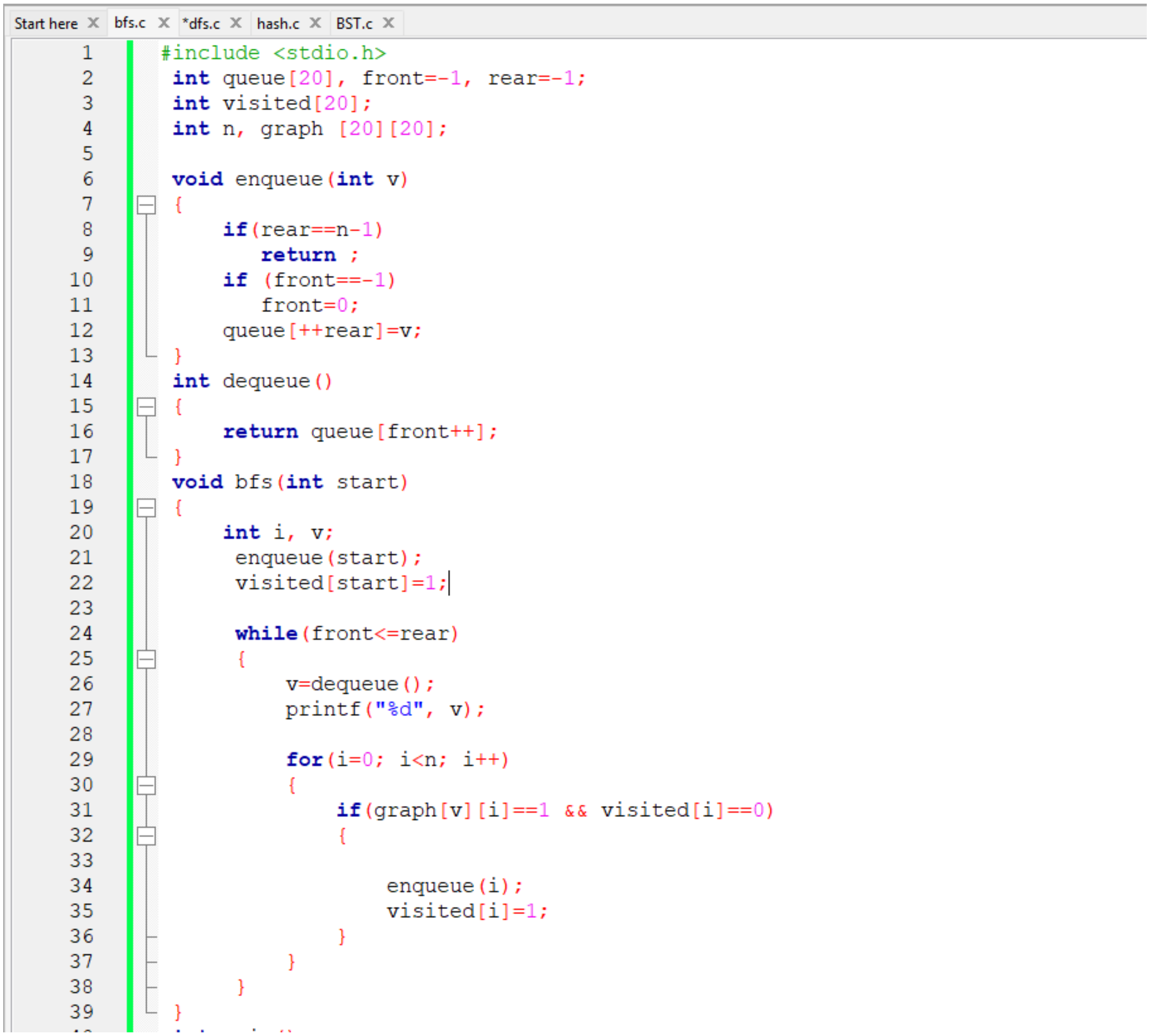
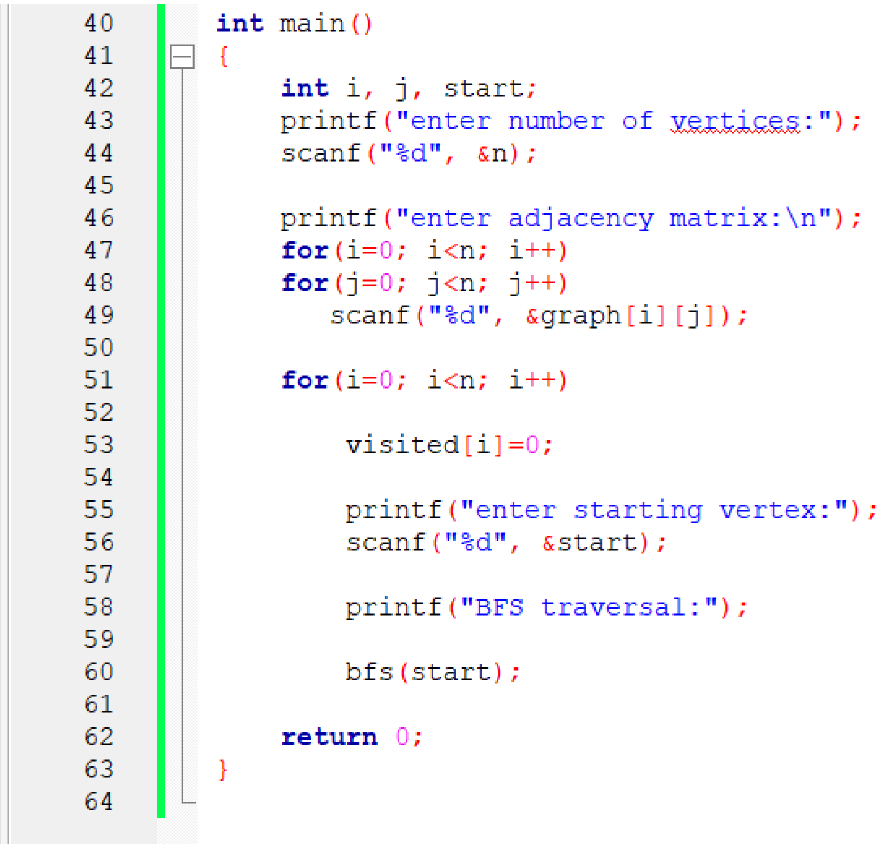
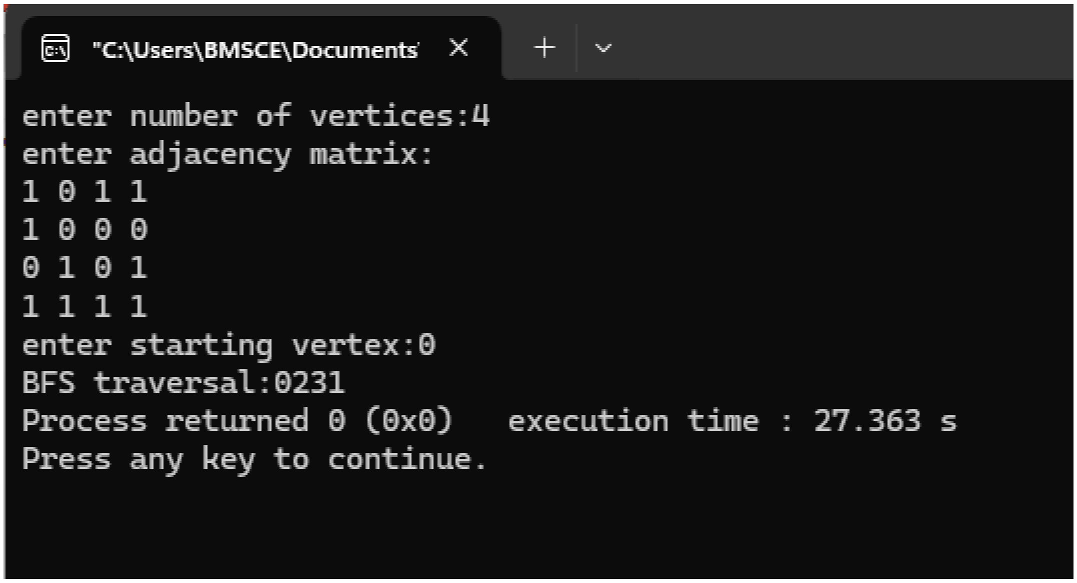
DS WEEK-10



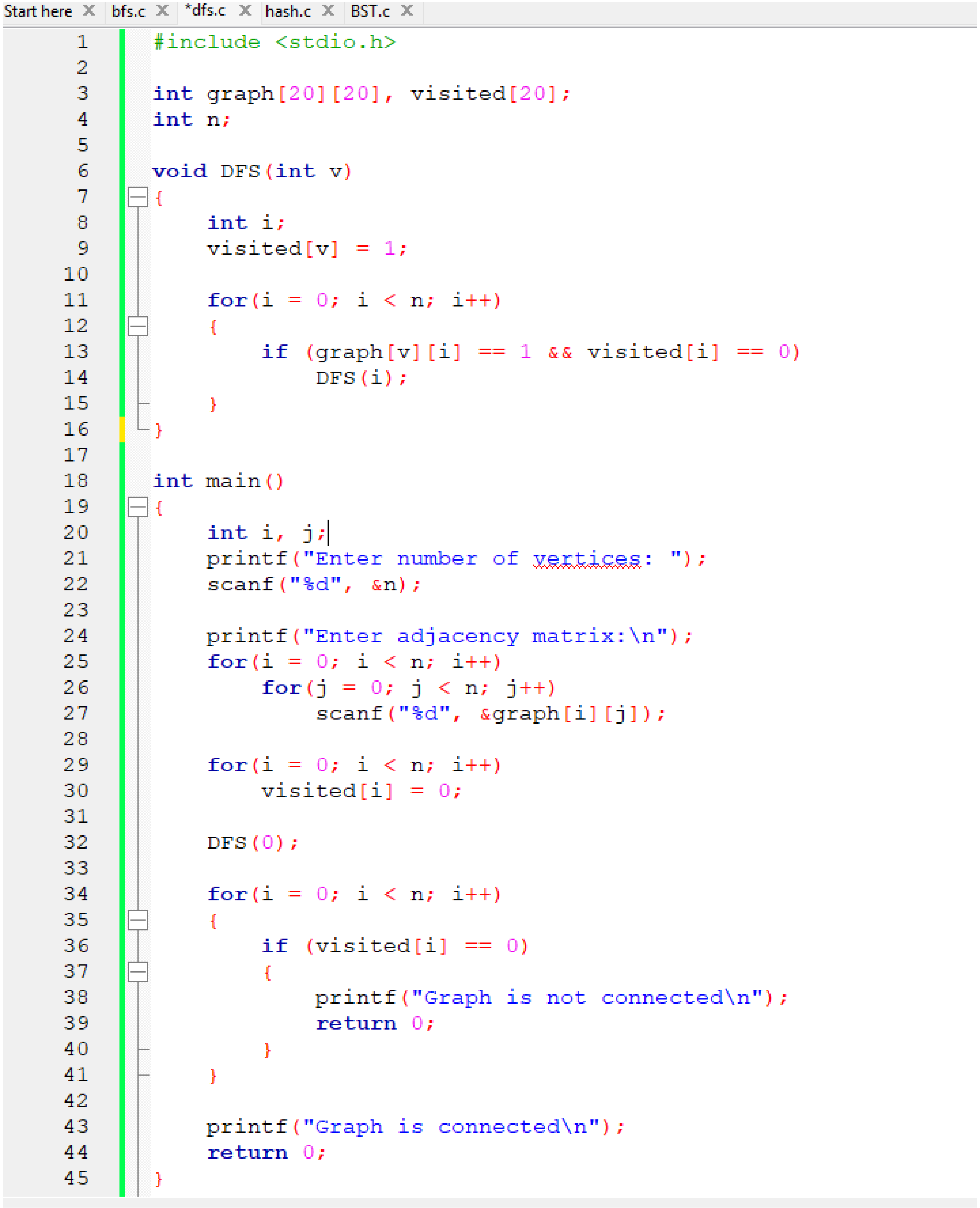
```c
#include <stdio.h>
int queue[20], front=-1, rear=-1;
int visited[20];
int n, graph [20][20];

void enqueue(int v)
{
    if(rear==n-1)
       return ;
    if (front==-1)
       front=0;
    queue[++rear]=v;
}
int dequeue()
{
    return queue[front++];
}
void bfs(int start)
{
    int i, v;
     enqueue(start);
     visited[start]=1;

     while(front<=rear)
     {
         v=dequeue();
         printf("%d", v);

         for(i=0; i<n; i++)
         {
             if(graph[v][i]==1 && visited[i]==0)
             {

                 enqueue(i);
                 visited[i]=1;
             }
         }
     }
}
```



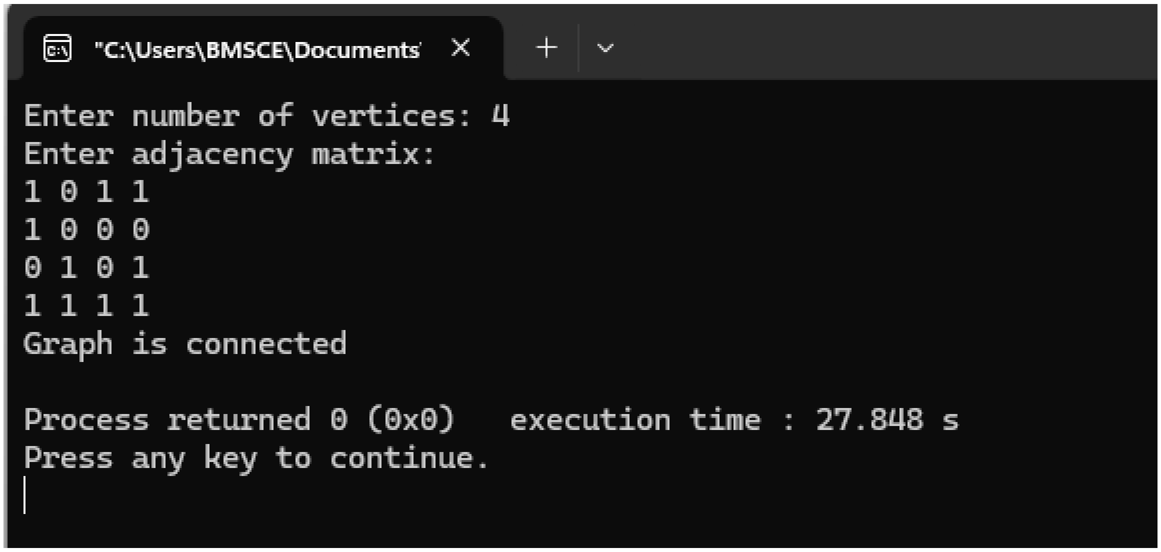
```
int main()
{
    int i, j, start;
    printf("enter number of vertices:");
    scanf("%d", &n);

    printf("enter adjacency matrix:\n");
    for(i=0; i<n; i++)
    for(j=0; j<n; j++)
       scanf("%d", &graph[i][j]);

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

        visited[i]=0;

        printf("enter starting vertex:");
        scanf("%d", &start);

        printf("BFS traversal:");

        bfs(start);

    return 0;
}
```



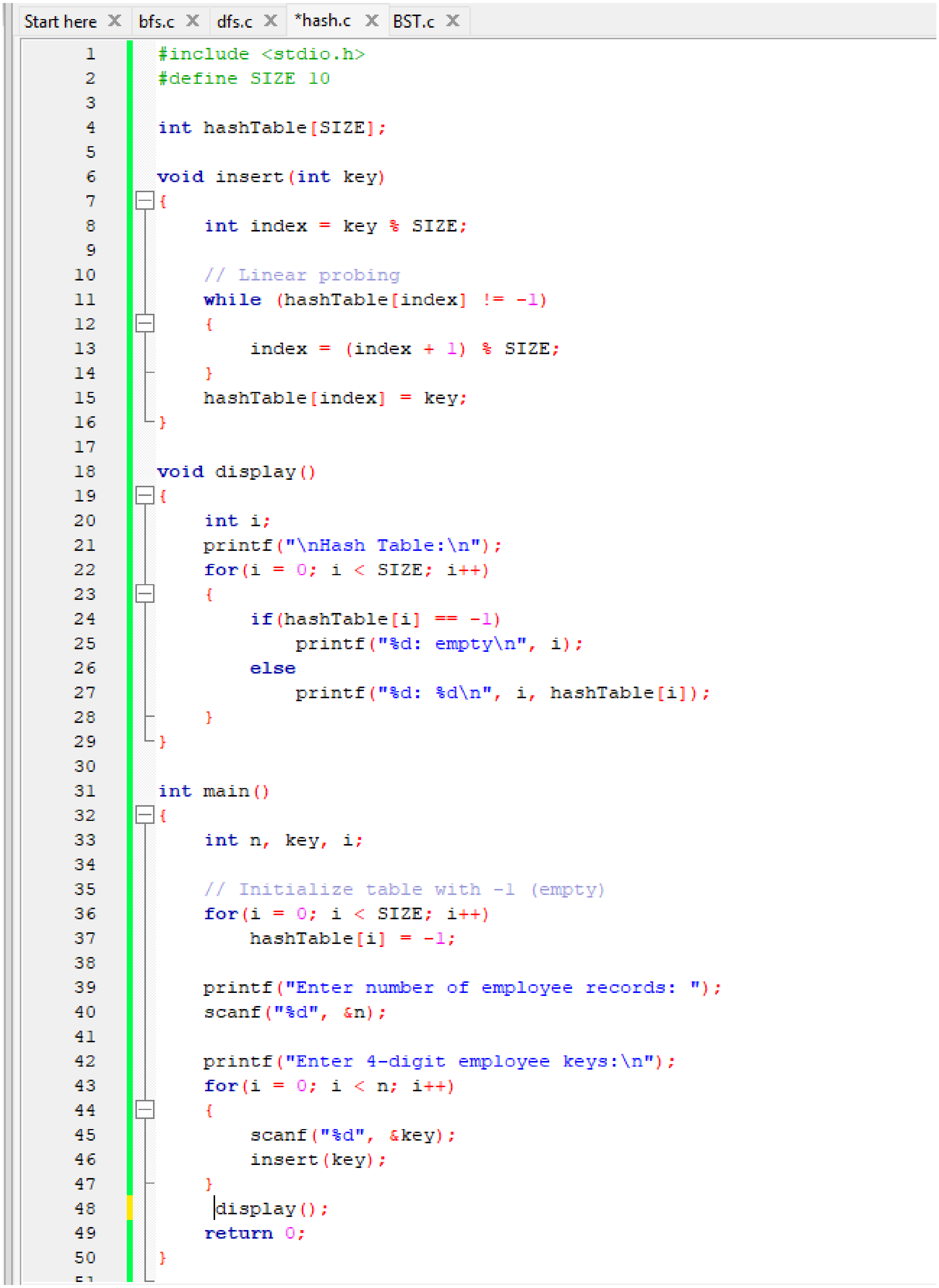
```
enter number of vertices:4
enter adjacency matrix:
1 0 1 1
1 0 0 0
0 1 0 1
1 1 1 1
enter starting vertex:0
BFS traversal:0231
Process returned 0 (0x0)   execution time : 27.363 s
Press any key to continue.
```



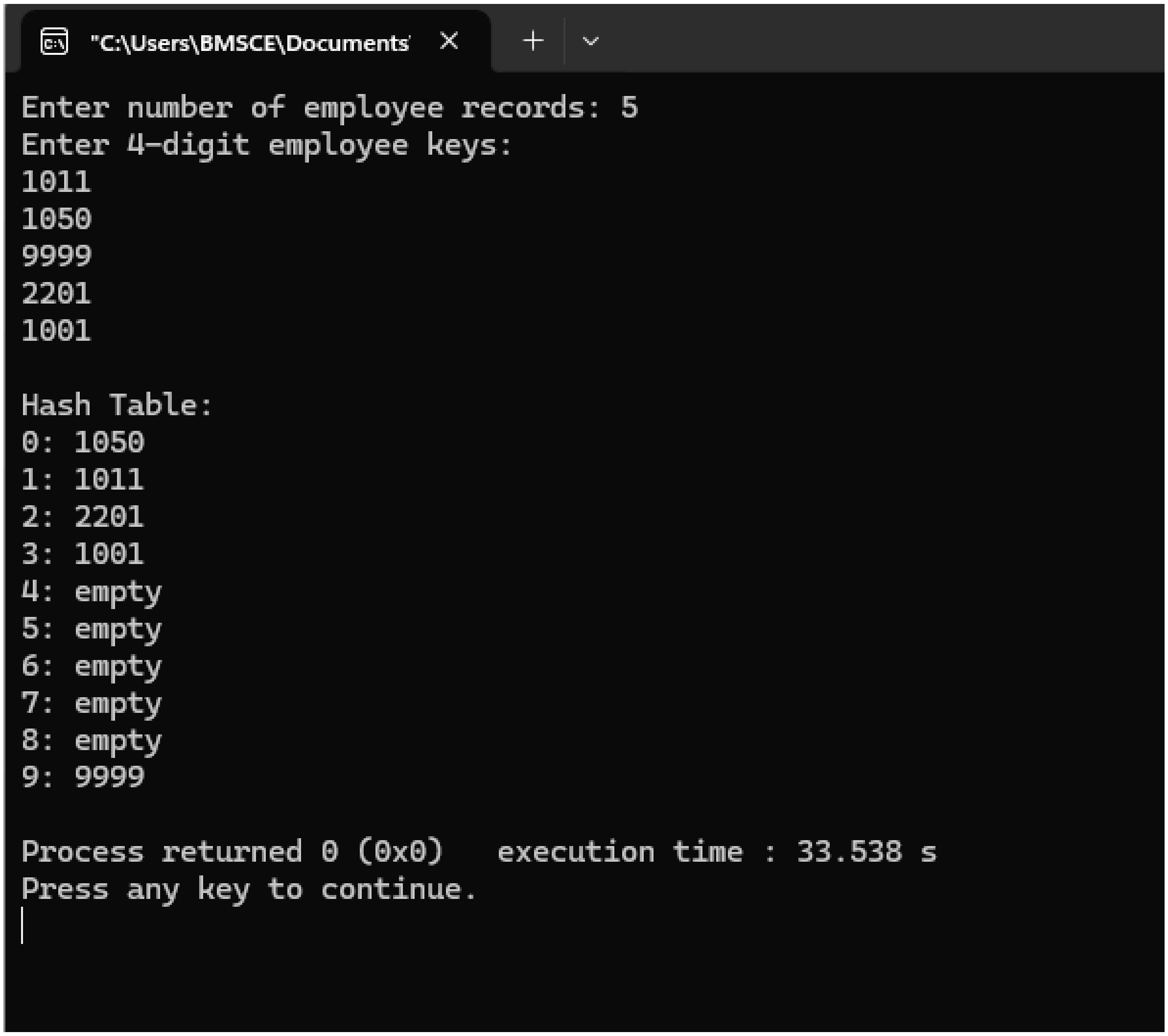
```c
#include <stdio.h>

int graph[20][20], visited[20];
int n;

void DFS(int v)
{
    int i;
    visited[v] = 1;

    for(i = 0; i < n; i++)
    {
        if (graph[v][i] == 1 && visited[i] == 0)
            DFS(i);
    }
}

int main()
{
    int i, j;
    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", &graph[i][j]);

    for(i = 0; i < n; i++)
        visited[i] = 0;

    DFS(0);

    for(i = 0; i < n; i++)
    {
        if (visited[i] == 0)
        {
            printf("Graph is not connected\n");
            return 0;
        }
    }

    printf("Graph is connected\n");
    return 0;
}
```



```
Enter number of vertices: 4
Enter adjacency matrix:
1 0 1 1
1 0 0 0
0 1 0 1
1 1 1 1
Graph is connected

Process returned 0 (0x0)   execution time : 27.848 s
Press any key to continue.
```



```c
#include <stdio.h>
#define SIZE 10

int hashTable[SIZE];

void insert(int key)
{
    int index = key % SIZE;

    // Linear probing
    while (hashTable[index] != -1)
    {
        index = (index + 1) % SIZE;
    }
    hashTable[index] = key;
}

void display()
{
    int i;
    printf("\nHash Table:\n");
    for(i = 0; i < SIZE; i++)
    {
        if(hashTable[i] == -1)
            printf("%d: empty\n", i);
        else
            printf("%d: %d\n", i, hashTable[i]);
    }
}

int main()
{
    int n, key, i;

    // Initialize table with -1 (empty)
    for(i = 0; i < SIZE; i++)
        hashTable[i] = -1;

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

    printf("Enter 4-digit employee keys:\n");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &key);
        insert(key);
    }
    display();
    return 0;
}
```



```
Enter number of employee records: 5
Enter 4-digit employee keys:
1011
1050
9999
2201
1001

Hash Table:
0: 1050
1: 1011
2: 2201
3: 1001
4: empty
5: empty
6: empty
7: empty
8: empty
9: 9999

Process returned 0 (0x0)   execution time : 33.538 s
Press any key to continue.
```