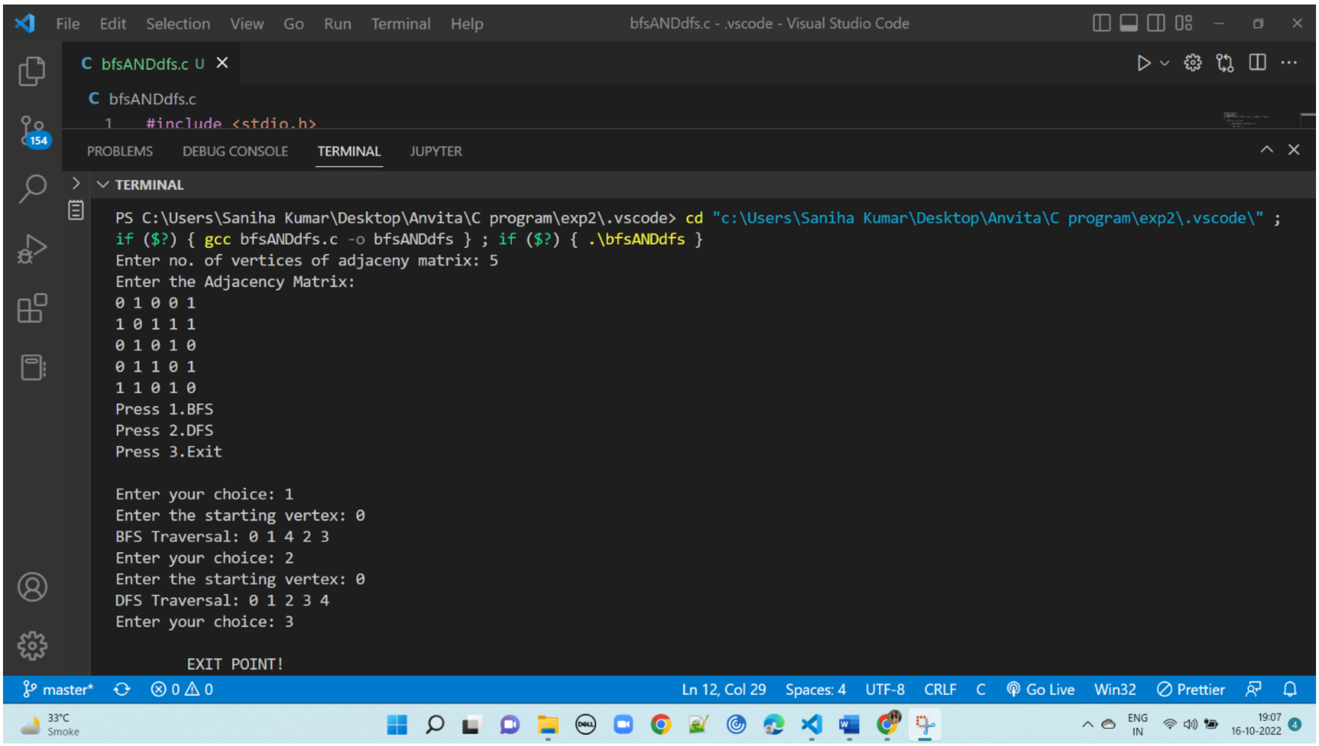
ANVITA KUMAR
C-22
Roll No.:- 2104097

```
//WAP to implement BFS and DFS in binary tree
#include <stdio.h>
#include <conio.h>
int adj[30][30], n;
void BFS(int front, int rear, int vis[], int queue[], int start)
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (adj[start][i] != 0 && vis[i] != 1)
        {
            rear = rear + 1;
            queue[rear] = i;
            vis[i] = 1;
            printf("%d ", i);
        }
    }
    front = front + 1;
    if (front <= rear)
        BFS(front, rear, vis, queue, queue[front]);
}
void DFS(int vis[], int start)
{
    int j;
    for (j = 0; j < n; j++)
    {
        if (vis[j] == 0 && adj[start][j] != 0)
        {
            vis[j] = 1;
            printf("%d ", j);
            DFS(vis, j);
        }
    }
}
int main()
{
    int choice, v;
    int front = -1, rear = -1;
    int queue[10], vis1[10], vis2[10] = {0};
    printf("Enter no. of vertices of adjaceny matrix: ");
    scanf("%d", &n);
    printf("Enter the 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++)
    {
        vis1[i] = 0;
    }
    printf("Press 1.BFS\n");
    printf("Press 2.DFS\n");
    printf("Press 3.Exit\n");
    do
    {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the starting vertex: ");
            scanf("%d", &v);
            front = 0;
            rear = 0;
            queue[rear] = v;
            vis1[v] = 1;
            printf("BFS Traversal: ");
            printf("%d ", v);
            BFS(front, rear, vis1, queue, v);
            break;
        case 2:
            printf("Enter the starting vertex: ");
            scanf("%d", &v);
            printf("DFS Traversal: ");
            vis2[v] = 1;
            printf("%d ", v);
            DFS(vis2, v);
            break;
        case 3:
            printf("\n\tEXIT POINT!");
        }
    } while (choice != 3);
    return 0;
}
```

ANVITA KUMAR
C-22
Roll No.:- 2104097


File Edit Selection View Go Run Terminal Help
bfsANDdfs.c - .vscode - Visual Studio Code
bfsANDdfs.c U
bfsANDdfs.c
1 #include <stdio.h>
PROBLEMS DEBUG CONSOLE TERMINAL JUPYTER
TERMINAL
master* ⊗ 0 ⚠ 0
Ln 12, Col 29 Spaces: 4 UTF-8 CRLF C Go Live Win32 Prettier
33°C Smoke
ENG IN 19:07 16-10-2022

```
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode> cd "c:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode\" ;
if ($?) { gcc bfsANDdfs.c -o bfsANDdfs } ; if ($?) { .\bfsANDdfs }
Enter no. of vertices of adjaceny matrix: 5
Enter the Adjacency Matrix:
0 1 0 0 1
1 0 1 1 1
0 1 0 1 0
0 1 1 0 1
1 1 0 1 0
Press 1.BFS
Press 2.DFS
Press 3.Exit

Enter your choice: 1
Enter the starting vertex: 0
BFS Traversal: 0 1 4 2 3
Enter your choice: 2
Enter the starting vertex: 0
DFS Traversal: 0 1 2 3 4
Enter your choice: 3

        EXIT POINT!
```

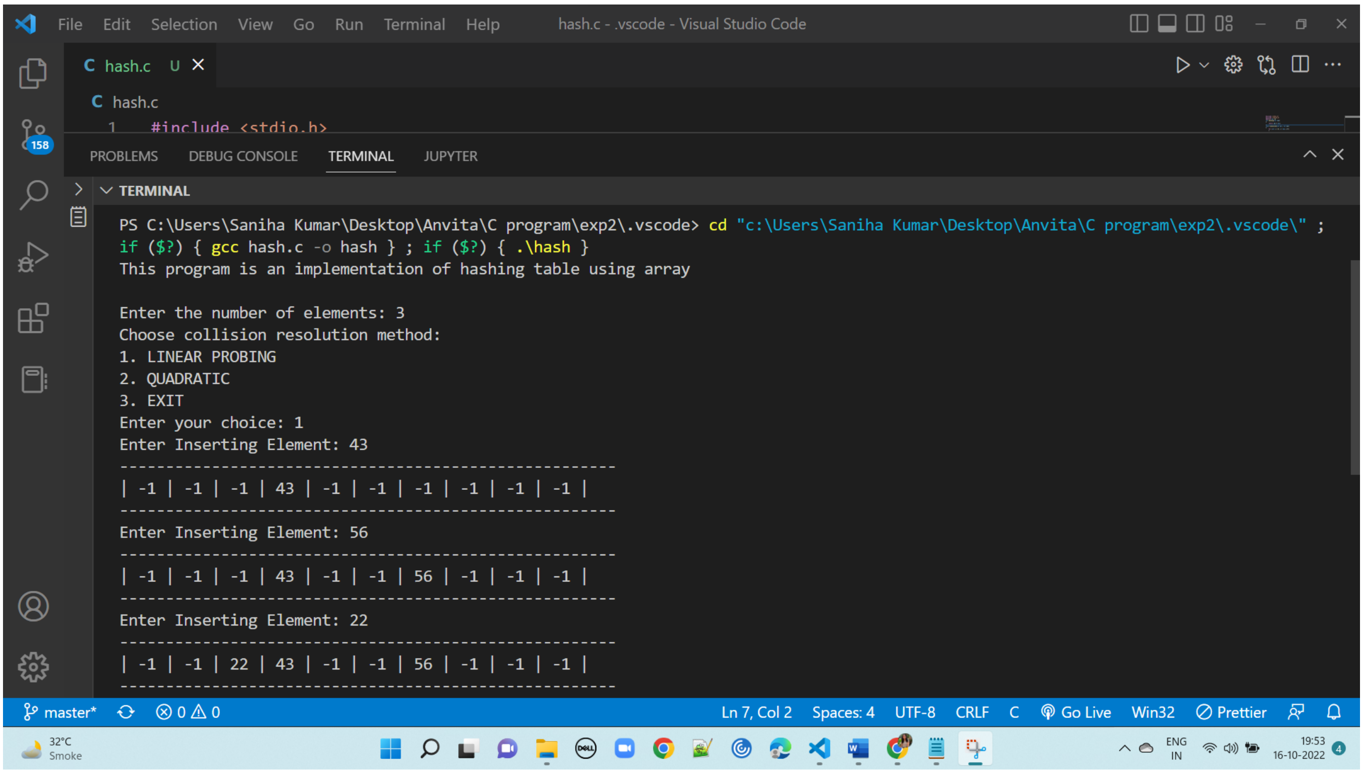
ANVITA KUMAR
C-22
Roll No.:- 2104097

```
//WAP to implement hashing table using array
#include <stdio.h>
#include <stdlib.h>
#define max 10
int hashing(int val)
{
   return val % max;
}
void linearprob(int a[], int val)
{
   for (int i = 0; i < max; i++)
   {
      int code = hashing(hashing(val) + i);
      if (a[code] == -1)
      {
         a[code] = val;
         break;
      }
   }
}
void quadprob(int a[], int val)
{
   for (int i = 0; i < max; i++)
   {
      int code = hashing(hashing(val) + i * i);
      if (a[code] == -1)
      {
         a[code] = val;
         break;
      }
   }
```

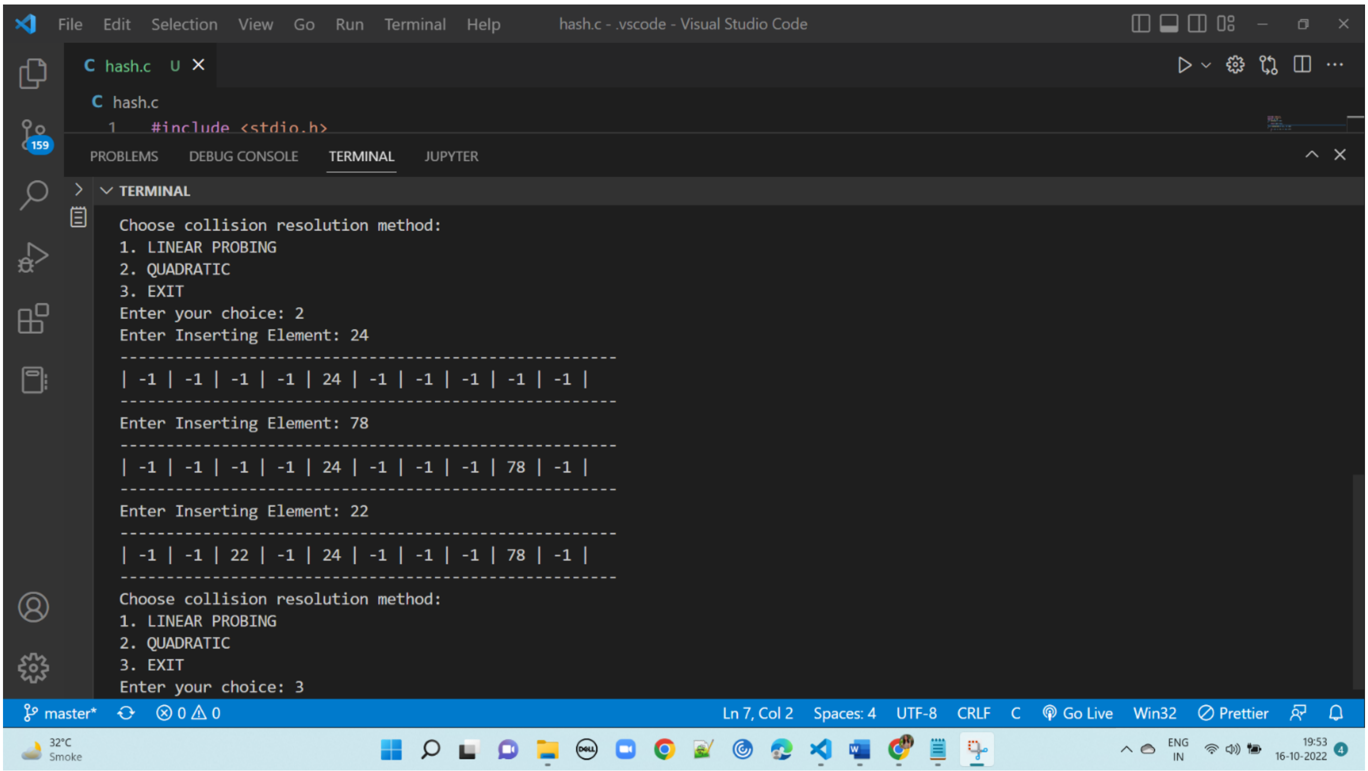
```
}

void display(int a[])

{

   printf("-----------------------------------------------------\n");

   for (int i = 0; i < max; i++)

   {

      printf("| %d ", a[i]);

   }

   printf("|\n-----------------------------------------------------\n");

}

void create(int a[])

{

   for (int i = 0; i < max; i++)

   {

      a[i] = -1;

   }

}

int main()

{

   int val, choice, n, a[max];

   printf("This program is an implementation of hashing table using array\n\n");

   printf("Enter the number of elements: ");

   scanf("%d", &n);

   do

   {

      create(a);

      printf("Choose collision resolution method:\n");

      printf("1. LINEAR PROBING\n2. QUADRATIC\n3. EXIT\n");

      printf("Enter your choice: ");

      scanf("%d", &choice);

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

```
        {
            printf("Enter Inserting Element: ");
            scanf("%d", &val);
            switch (choice)
            {
            case 1:
                linearprob(a, val);
                display(a);
                break;
            case 2:
                quadprob(a, val);
                display(a);
                break;
            case 3:
                printf("\n\tEXIT POINT!");
                break;
            }
        }
    } while (choice != 3);
    return 0;
}
```

ANVITA KUMAR
C-22
Roll No.:- 2104097



```
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode> cd "c:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode\" ;
if ($?) { gcc hash.c -o hash } ; if ($?) { .\hash }
This program is an implementation of hashing table using array

Enter the number of elements: 3
Choose collision resolution method:
1. LINEAR PROBING
2. QUADRATIC
3. EXIT
Enter your choice: 1
Enter Inserting Element: 43
------------------------------------------------------
| -1 | -1 | -1 | 43 | -1 | -1 | -1 | -1 | -1 | -1 |
------------------------------------------------------
Enter Inserting Element: 56
------------------------------------------------------
| -1 | -1 | -1 | 43 | -1 | -1 | 56 | -1 | -1 | -1 |
------------------------------------------------------
Enter Inserting Element: 22
------------------------------------------------------
| -1 | -1 | 22 | 43 | -1 | -1 | 56 | -1 | -1 | -1 |
------------------------------------------------------
```



```
Choose collision resolution method:
1. LINEAR PROBING
2. QUADRATIC
3. EXIT
Enter your choice: 2
Enter Inserting Element: 24
------------------------------------------------------
| -1 | -1 | -1 | -1 | 24 | -1 | -1 | -1 | -1 | -1 |
------------------------------------------------------
Enter Inserting Element: 78
------------------------------------------------------
| -1 | -1 | -1 | -1 | 24 | -1 | -1 | -1 | 78 | -1 |
------------------------------------------------------
Enter Inserting Element: 22
------------------------------------------------------
| -1 | -1 | 22 | -1 | 24 | -1 | -1 | -1 | 78 | -1 |
------------------------------------------------------
Choose collision resolution method:
1. LINEAR PROBING
2. QUADRATIC
3. EXIT
Enter your choice: 3
```