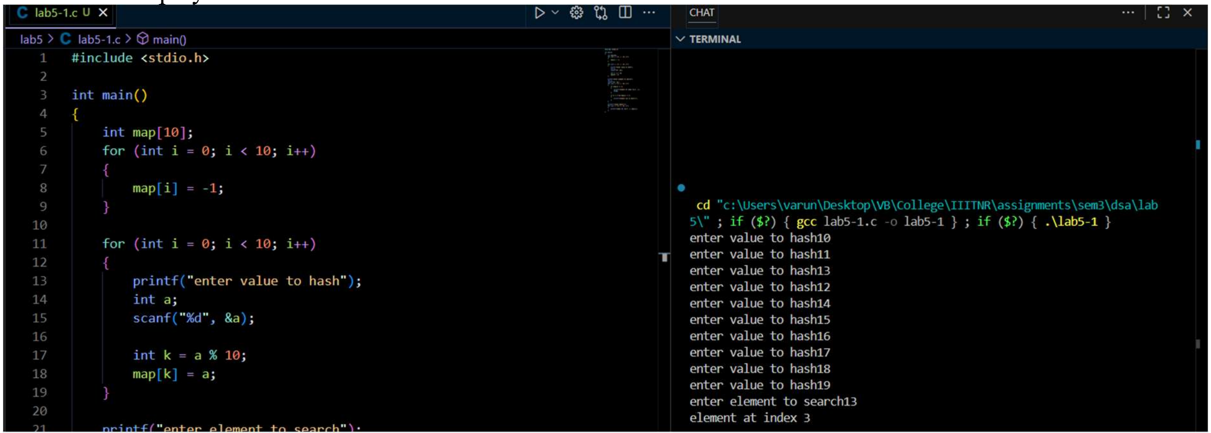
VARUN BHARGAVA – 241010282
DATA STRUCTURES TASK-5

Task 01: Hashing:

(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab5/lab5-1.c)
1. Implement a hash table for storing student roll numbers using the division method of hashing.
Use hash function: Index = Roll_no % table_size

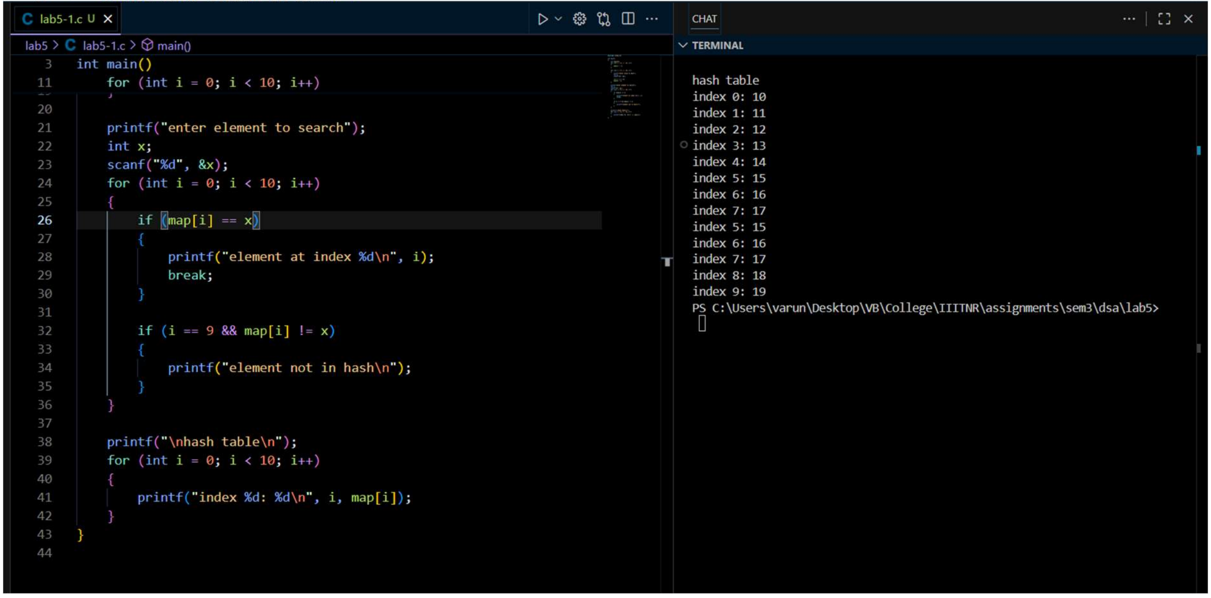
- Insert 10 roll numbers into the hash table (assume no collisions).
Roll no.: 10, 21, 32, 43, 54, 65, 76, 87, 98, 109

- Search for a given roll number and display its index.
- Display/Print the hash table.



```c
#include <stdio.h>

int main()
{
    int map[10];
    for (int i = 0; i < 10; i++)
    {
        map[i] = -1;
    }

    for (int i = 0; i < 10; i++)
    {
        printf("enter value to hash");
        int a;
        scanf("%d", &a);

        int k = a % 10;
        map[k] = a;
    }

    printf("enter element to search");
```

```
cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab
5\" ; if ($?) { gcc lab5-1.c -o lab5-1 } ; if ($?) { .\lab5-1 }
enter value to hash10
enter value to hash11
enter value to hash13
enter value to hash12
enter value to hash14
enter value to hash15
enter value to hash16
enter value to hash17
enter value to hash18
enter value to hash19
enter element to search13
element at index 3
```



```c
int main()
    for (int i = 0; i < 10; i++)

    printf("enter element to search");
    int x;
    scanf("%d", &x);
    for (int i = 0; i < 10; i++)
    {
        if (map[i] == x)
        {
            printf("element at index %d\n", i);
            break;
        }

        if (i == 9 && map[i] != x)
        {
            printf("element not in hash\n");
        }
    }

    printf("\nhash table\n");
    for (int i = 0; i < 10; i++)
    {
        printf("index %d: %d\n", i, map[i]);
    }
}
```

```
hash table
index 0: 10
index 1: 11
index 2: 12
index 3: 13
index 4: 14
index 5: 15
index 6: 16
index 7: 17
index 5: 15
index 6: 16
index 7: 17
index 8: 18
index 9: 19
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
```

Task 02: Chaining:

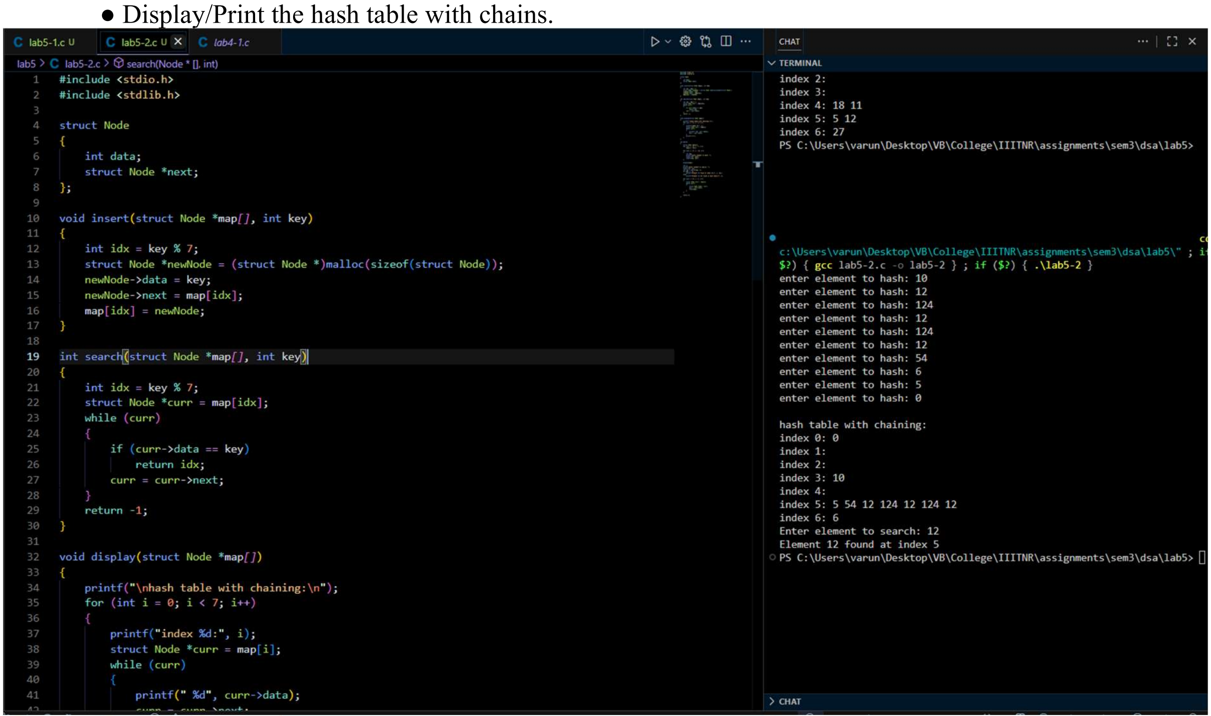
(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab5/lab5-2.c)

Use hash function : Index = Integer_keys % 7

- Use linked lists to handle collisions. If collision occurs insert at the beginning of the chain(Linked list).
- Insert 10 integer keys.

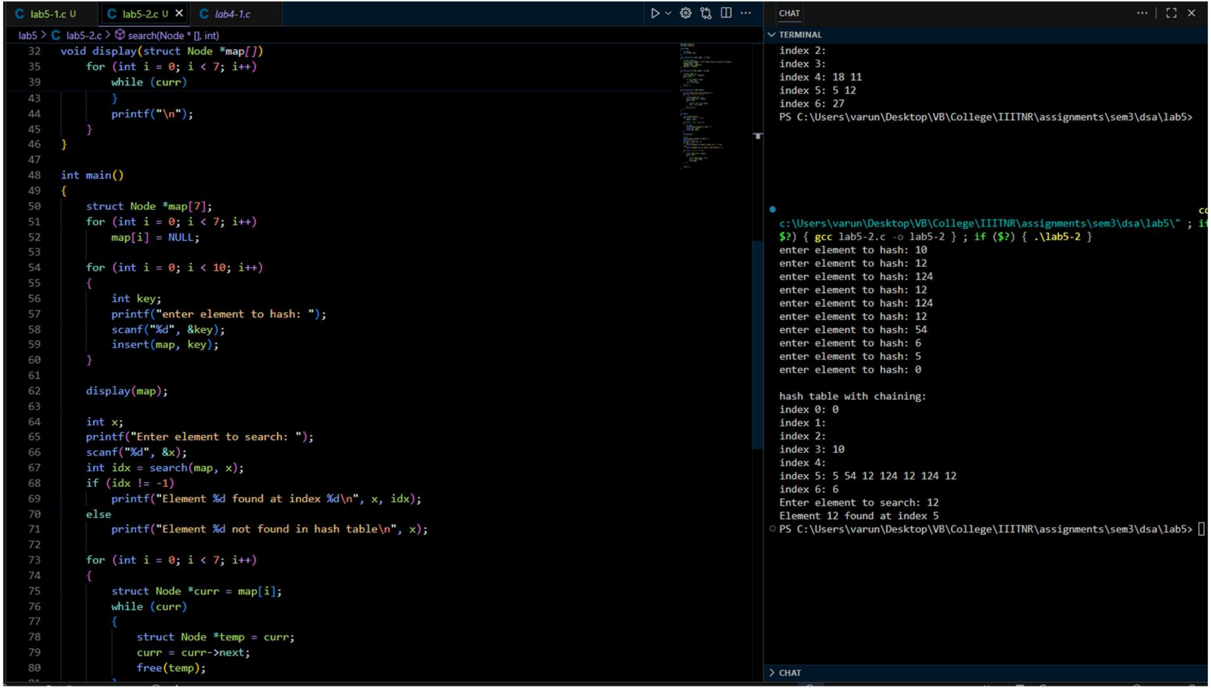
Integer keys : 15, 11, 27, 8, 12, 14, 5, 7, 18, 29

- Display/Print the hash table with chains.



```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};

void insert(struct Node *map[], int key)
{
    int idx = key % 7;
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = key;
    newNode->next = map[idx];
    map[idx] = newNode;
}

int search(struct Node *map[], int key)
{
    int idx = key % 7;
    struct Node *curr = map[idx];
    while (curr)
    {
        if (curr->data == key)
            return idx;
        curr = curr->next;
    }
    return -1;
}

void display(struct Node *map[])
{
    printf("\nhash table with chaining:\n");
    for (int i = 0; i < 7; i++)
    {
        printf("index %d:", i);
        struct Node *curr = map[i];
        while (curr)
        {
            printf(" %d", curr->data);
```

```
index 2:
index 3:
index 4: 18 11
index 5: 5 12
index 6: 27
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>

                                                                              co
c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5\" ; i
$?) { gcc lab5-2.c -o lab5-2 } ; if ($?) { .\lab5-2 }
enter element to hash: 10
enter element to hash: 12
enter element to hash: 124
enter element to hash: 12
enter element to hash: 124
enter element to hash: 12
enter element to hash: 54
enter element to hash: 6
enter element to hash: 5
enter element to hash: 0

hash table with chaining:
index 0: 0
index 1:
index 2:
index 3: 10
index 4:
index 5: 5 54 12 124 12 124 12
index 6: 6
Enter element to search: 12
Element 12 found at index 5
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
```



```c
void display(struct Node *map[])
    for (int i = 0; i < 7; i++)
        while (curr)
        }
        printf("\n");
    }
}

int main()
{
    struct Node *map[7];
    for (int i = 0; i < 7; i++)
        map[i] = NULL;

    for (int i = 0; i < 10; i++)
    {
        int key;
        printf("enter element to hash: ");
        scanf("%d", &key);
        insert(map, key);
    }

    display(map);

    int x;
    printf("Enter element to search: ");
    scanf("%d", &x);
    int idx = search(map, x);
    if (idx != -1)
        printf("Element %d found at index %d\n", x, idx);
    else
        printf("Element %d not found in hash table\n", x);

    for (int i = 0; i < 7; i++)
    {
        struct Node *curr = map[i];
        while (curr)
        {
            struct Node *temp = curr;
            curr = curr->next;
            free(temp);
```

```
index 2:
index 3:
index 4: 18 11
index 5: 5 12
index 6: 27
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>

                                                                              co
c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5\" ; i
$?) { gcc lab5-2.c -o lab5-2 } ; if ($?) { .\lab5-2 }
enter element to hash: 10
enter element to hash: 12
enter element to hash: 124
enter element to hash: 12
enter element to hash: 124
enter element to hash: 12
enter element to hash: 54
enter element to hash: 6
enter element to hash: 5
enter element to hash: 0

hash table with chaining:
index 0: 0
index 1:
index 2:
index 3: 10
index 4:
index 5: 5 54 12 124 12 124 12
index 6: 6
Enter element to search: 12
Element 12 found at index 5
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
```

Task 03: Linear Probing:

(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab5/lab5-3.c)

Use hash function : Index = Integer_keys % 10

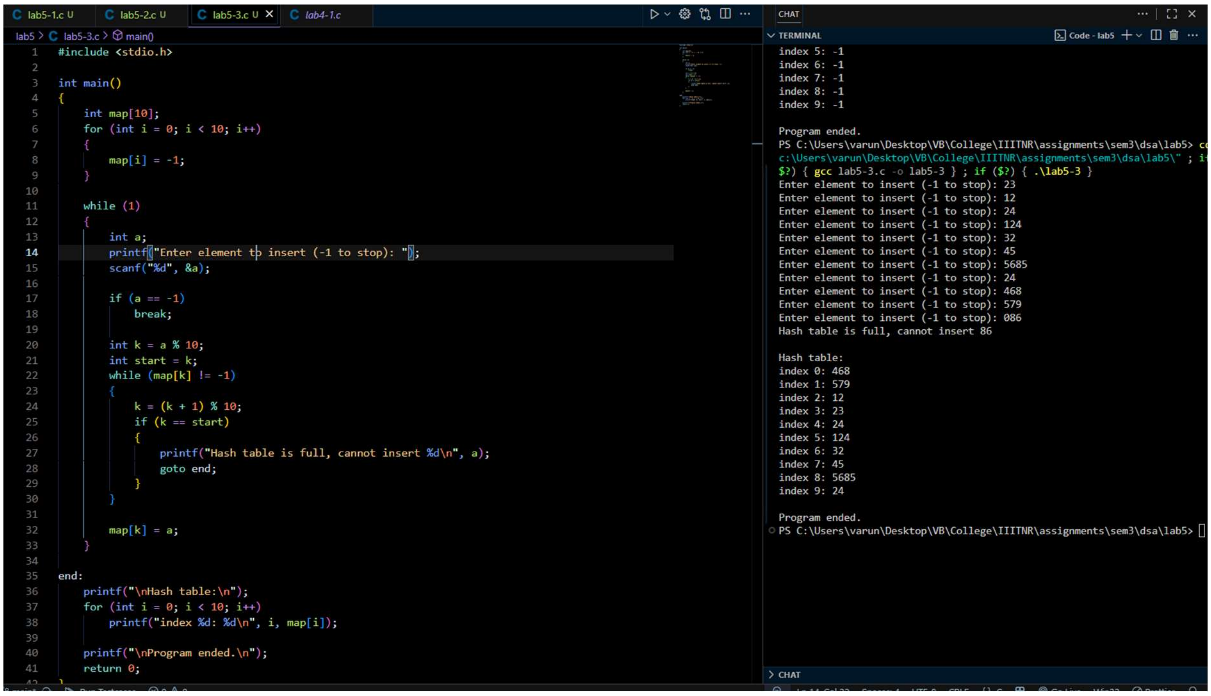
- Insert 10 integer keys using linear probing.

Integer keys : 23, 43, 13, 27, 39, 54, 31, 72, 18, 29

- Show how collisions are resolved by checking the next available slot.

1. Resolve collisions by checking the next available slot in a linear sequence.
2. If a slot is occupied, move to the next slot (index + 1) % 10.

- Print the final hash table.



```c
#include <stdio.h>

int main()
{
    int map[10];
    for (int i = 0; i < 10; i++)
    {
        map[i] = -1;
    }

    while (1)
    {
        int a;
        printf("Enter element to insert (-1 to stop): ");
        scanf("%d", &a);

        if (a == -1)
            break;

        int k = a % 10;
        int start = k;
        while (map[k] != -1)
        {
            k = (k + 1) % 10;
            if (k == start)
            {
                printf("Hash table is full, cannot insert %d\n", a);
                goto end;
            }
        }

        map[k] = a;
    }

end:
    printf("\nHash table:\n");
    for (int i = 0; i < 10; i++)
        printf("index %d: %d\n", i, map[i]);

    printf("\nProgram ended.\n");
    return 0;
}
```

```
index 5: -1
index 6: -1
index 7: -1
index 8: -1
index 9: -1

Program ended.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5> co
c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5\" ; i
$?) { gcc lab5-3.c -o lab5-3 } ; if ($?) { .\lab5-3 }
Enter element to insert (-1 to stop): 23
Enter element to insert (-1 to stop): 12
Enter element to insert (-1 to stop): 24
Enter element to insert (-1 to stop): 124
Enter element to insert (-1 to stop): 32
Enter element to insert (-1 to stop): 45
Enter element to insert (-1 to stop): 5685
Enter element to insert (-1 to stop): 24
Enter element to insert (-1 to stop): 468
Enter element to insert (-1 to stop): 579
Enter element to insert (-1 to stop): 086
Hash table is full, cannot insert 86

Hash table:
index 0: 468
index 1: 579
index 2: 12
index 3: 23
index 4: 24
index 5: 124
index 6: 32
index 7: 45
index 8: 5685
index 9: 24

Program ended.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
```

Task 04: Quadratic Probing:

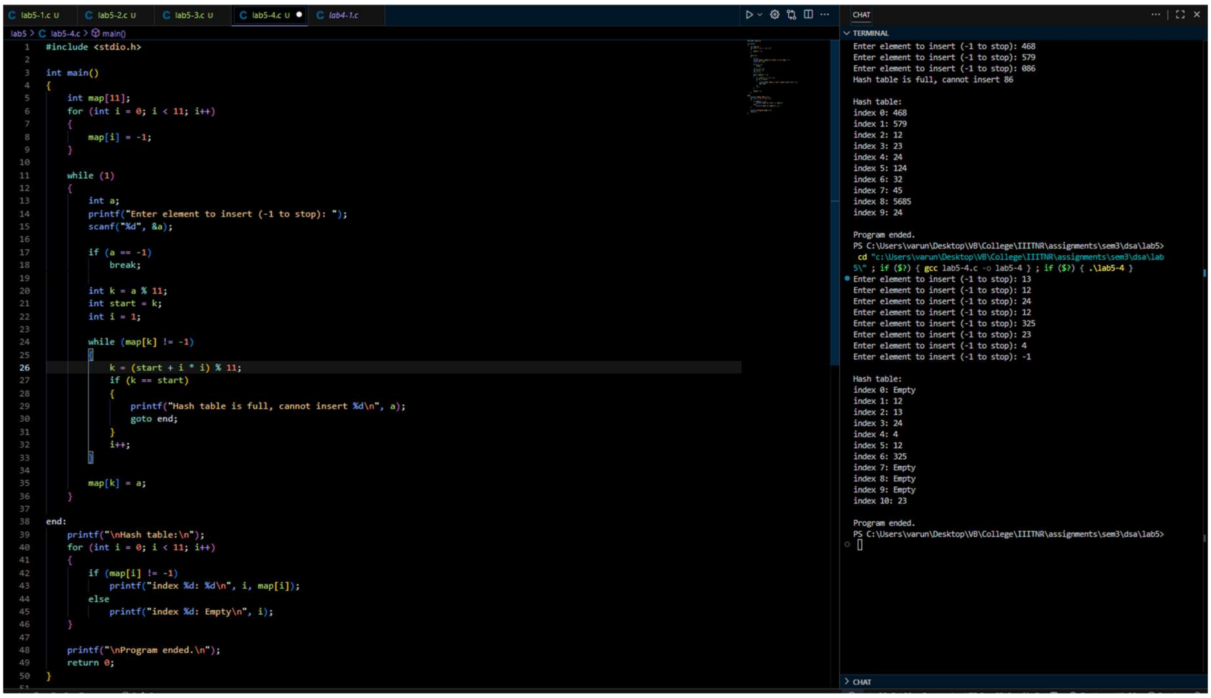
(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab5/lab5-4.c)

Use hash function : Index = Integer_keys % 11

- Insert 10 integer keys using quadratic probing.

  Integer keys : 19, 27, 36, 10, 64, 29, 20, 55, 39, 75

- Resolve collisions using quadratic steps.
  If the index is occupied, try (h(Integer_keys) + i^2) % 11 increment i = 1, 2, 3, ...
  until an empty slot is found (where h is the hash function.)
- Print the final hash table.



```c
#include <stdio.h>

int main()
{
    int map[11];
    for (int i = 0; i < 11; i++)
    {
        map[i] = -1;
    }

    while (1)
    {
        int a;
        printf("Enter element to insert (-1 to stop): ");
        scanf("%d", &a);

        if (a == -1)
            break;

        int k = a % 11;
        int start = k;
        int i = 1;

        while (map[k] != -1)
        {
            k = (start + i * i) % 11;
            if (k == start)
            {
                printf("Hash table is full, cannot insert %d\n", a);
                goto end;
            }
            i++;
        }

        map[k] = a;
    }

end:
    printf("\nHash table:\n");
    for (int i = 0; i < 11; i++)
    {
        if (map[i] != -1)
            printf("index %d: %d\n", i, map[i]);
        else
            printf("index %d: Empty\n", i);
    }

    printf("\nProgram ended.\n");
    return 0;
}
```

```
Enter element to insert (-1 to stop): 468
Enter element to insert (-1 to stop): 579
Enter element to insert (-1 to stop): 086
Hash table is full, cannot insert 86

Hash table:
index 0: 468
index 1: 579
index 2: 12
index 3: 23
index 4: 24
index 5: 124
index 6: 32
index 7: 45
index 8: 5685
index 9: 24

Program ended.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
 cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab
5\" ; if ($?) { gcc lab5-4.c -o lab5-4 } ; if ($?) { .\lab5-4 }
Enter element to insert (-1 to stop): 13
Enter element to insert (-1 to stop): 12
Enter element to insert (-1 to stop): 24
Enter element to insert (-1 to stop): 12
Enter element to insert (-1 to stop): 325
Enter element to insert (-1 to stop): 23
Enter element to insert (-1 to stop): 4
Enter element to insert (-1 to stop): -1

Hash table:
index 0: Empty
index 1: 12
index 2: 13
index 3: 24
index 4: 4
index 5: 12
index 6: 325
index 7: Empty
index 8: Empty
index 9: Empty
index 10: 23

Program ended.
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3\dsa\lab5>
```