DAY 15 – HASH TABLE

18.. Write a menu driven C program to implement hash table and the following collision resolution techniques-(i) Linear Probing (ii) Quadratic Probing (iii) Chaining

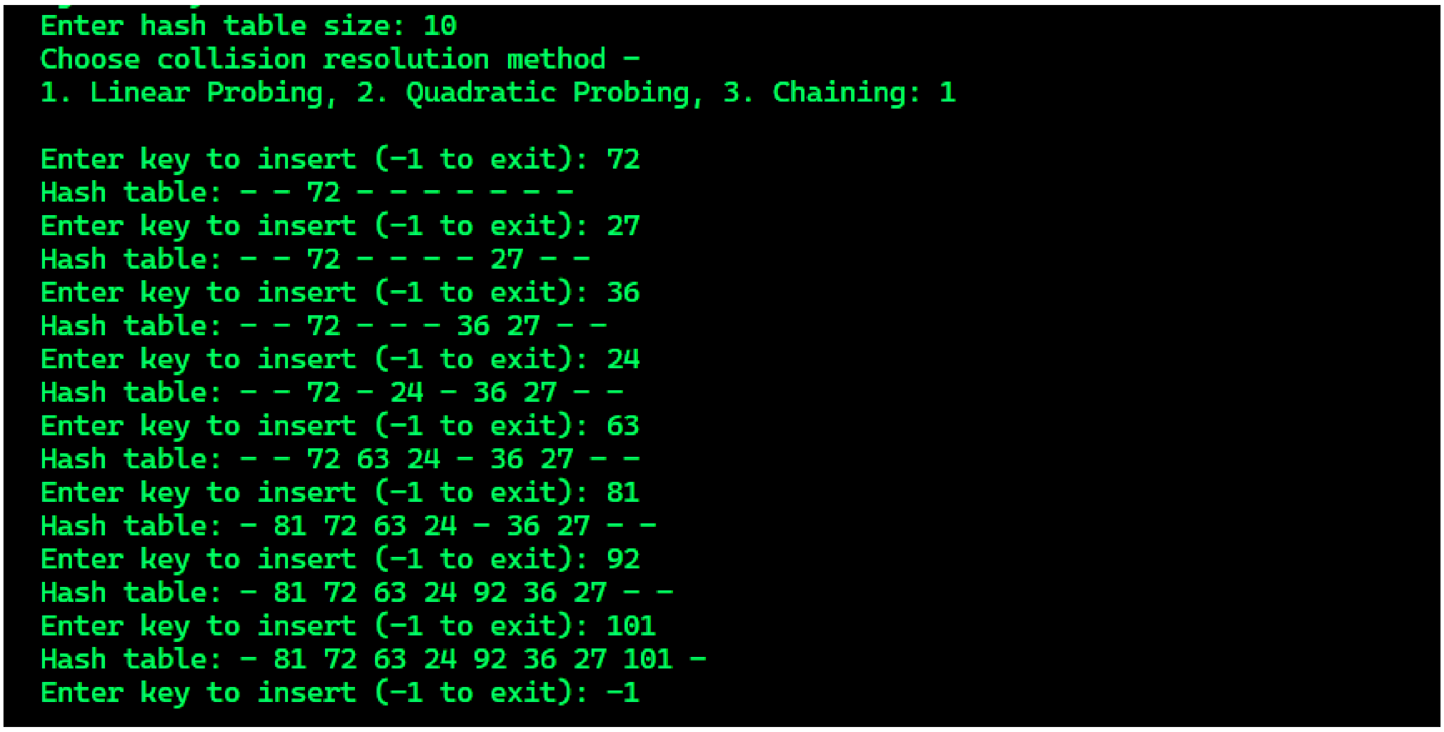
PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int lin_prob_array[MAX];
int quad_prob_array[MAX];
int size, c1, c2;
struct node
{
    int key;
    struct node *next;
}*chaining_array[MAX] = {NULL};
void display_lin_quad(int a[MAX])
{
    int i;
    printf("Hash table: ");
    for(i = 0; i < size; i++)
    {
        if(a[i] != -1)
        printf("%d ", a[i]);
        else
        printf("- ");
    }
}
void display_chaining()
{
    int i;
    struct node *ptr;
    printf("Hash table:\n");
    for(i = 0; i < size; i++)
    {
        printf("%d - ", i);
        ptr = chaining_array[i];
        if(chaining_array[i] == NULL)
        printf("NULL");
        else
        {
            while(ptr != NULL)
            {
                printf("%d -> ", ptr -> key);
                ptr = ptr -> next;
            }
            printf("NULL");
        }
        printf("\n");
    }
}
void insert_lin_prob(int key)
{
    int i, hash, finalhash, flag = 1;
    for(i = 0; i < size; i++)
    {
        if(lin_prob_array[i] == -1)
        flag = 0;
    }
    if(flag == 1)
    {
        printf("\nHash table is full.");
        return;
    }
    finalhash = hash = key % size;
    if(lin_prob_array[hash] != -1)
    {
        for(i = 0; i < size; i++)
        {
            finalhash = (hash + i) % size;
            if(lin_prob_array[finalhash] == -1)
            break;
        }
    }
    lin_prob_array[finalhash] = key;
    if(key!= -1)
    display_lin_quad(lin_prob_array);
}
void insert_quad_prob(int key)
{
    int i, hash, finalhash, flag = 1;
    for(i = 0; i < size; i++)
    {
        if(lin_prob_array[i] == -1)
        flag = 0;
    }
    if(flag == 1)
    {
        printf("\nHash table is full.");
        return;
    }
    finalhash = hash = key % size;
    if(quad_prob_array[hash] != -1)
    {
        for(i = 0; i < size; i++)
        {
            finalhash = (hash + c1 * i + c2 * i * i) % size;
            if(quad_prob_array[finalhash] == -1)
                break;
        }
```

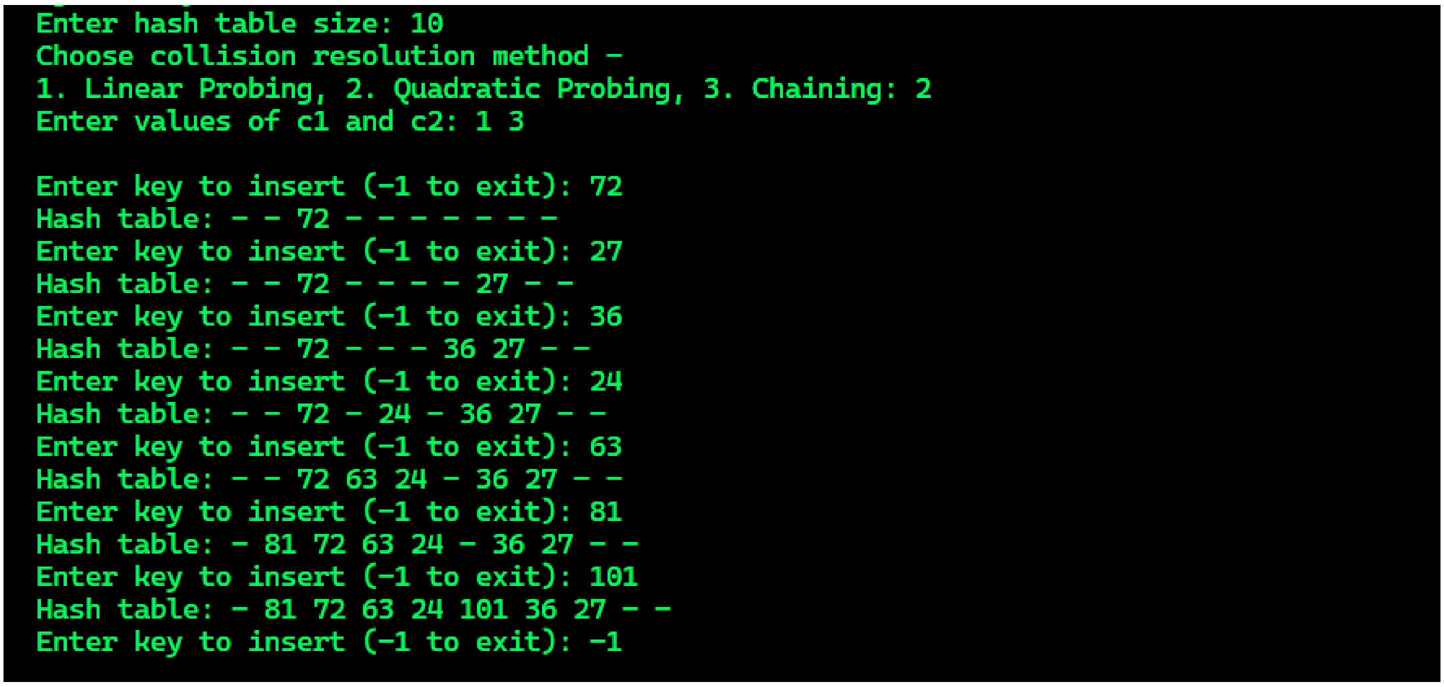
```c
    }
    quad_prob_array[finalhash] = key;
    if(key != -1)
        display_lin_quad(quad_prob_array);
}
void insert_chaining(int key)
{
    int i, hash;
    struct node *newnode, *ptr;
    hash = key % size;
    newnode = (struct node *)malloc(sizeof(struct node *));
    newnode -> key = key;
    newnode -> next = NULL;
    if(chaining_array[hash] == NULL)
        chaining_array[hash] = newnode;
    else
    {
        ptr = chaining_array[hash];
        while(ptr -> next != NULL)
        ptr = ptr -> next;
        ptr -> next = newnode;
    }
    if(key != -1)
        display_chaining();
}
void main()
{
    int ch, key,i;
    for(i=0;i<MAX;i++)
    {
        lin_prob_array[i]=-1;
        quad_prob_array[i]=-1;
    }
    printf("Enter hash table size: ");
    scanf("%d", &size);
    printf("Choose collision resolution method -\n");
    printf("1. Linear Probing, 2. Quadratic Probing, 3. Chaining: ");
    scanf("%d", &ch);
    if(ch == 2)
    {
        printf("Enter values of c1 and c2: ");
        scanf("%d%d", &c1, &c2);
    }
    do
    {
        printf("\nEnter key to insert (-1 to exit): ");
        scanf("%d", &key);
        switch(ch)
        {
        case 1: insert_lin_prob(key);
                break;
        case 2: insert_quad_prob(key);
                break;
        case 3: insert_chaining(key);
```

```
                    break;
            }
    }while(key != -1);
}
```

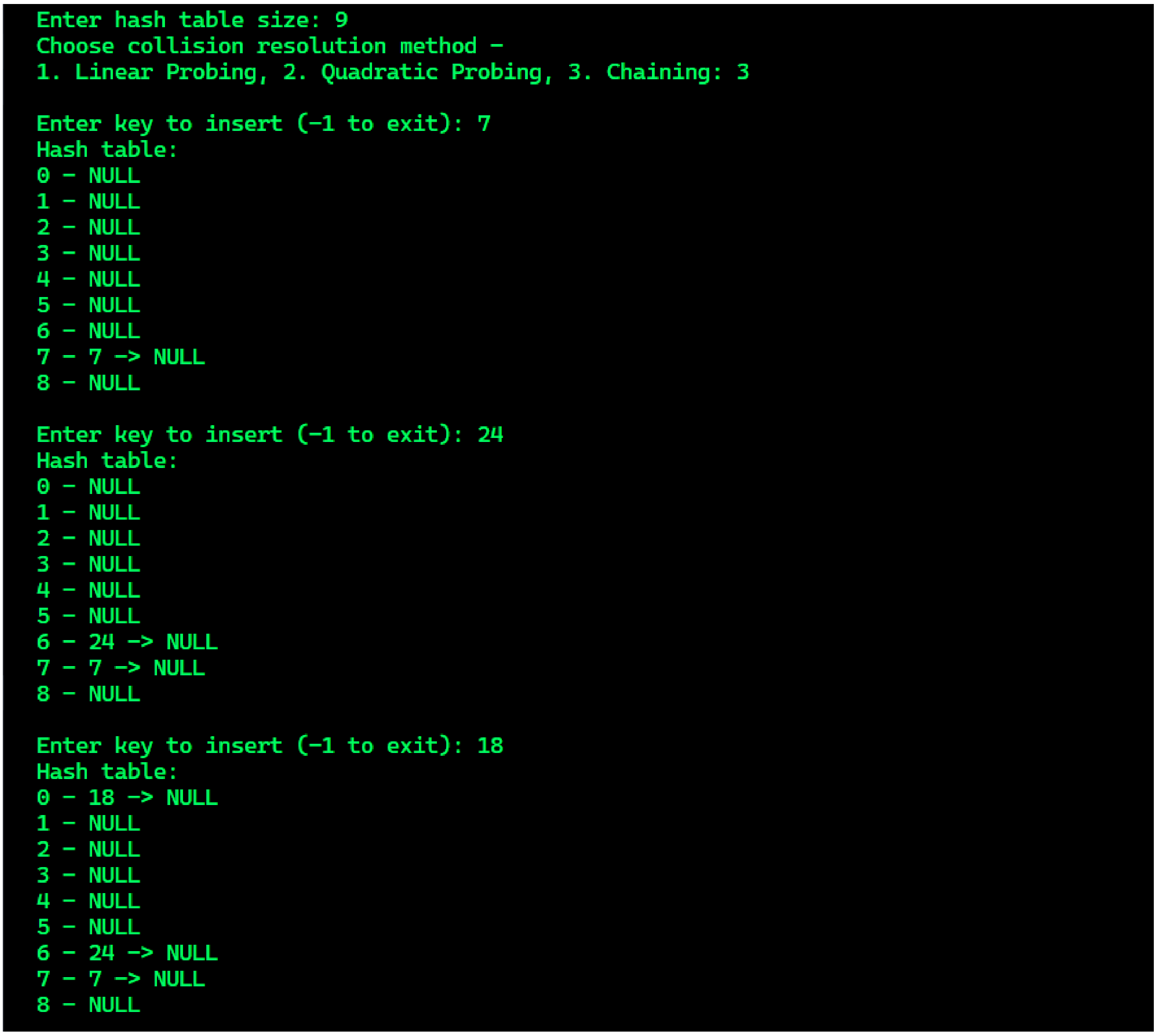
OUTPUT



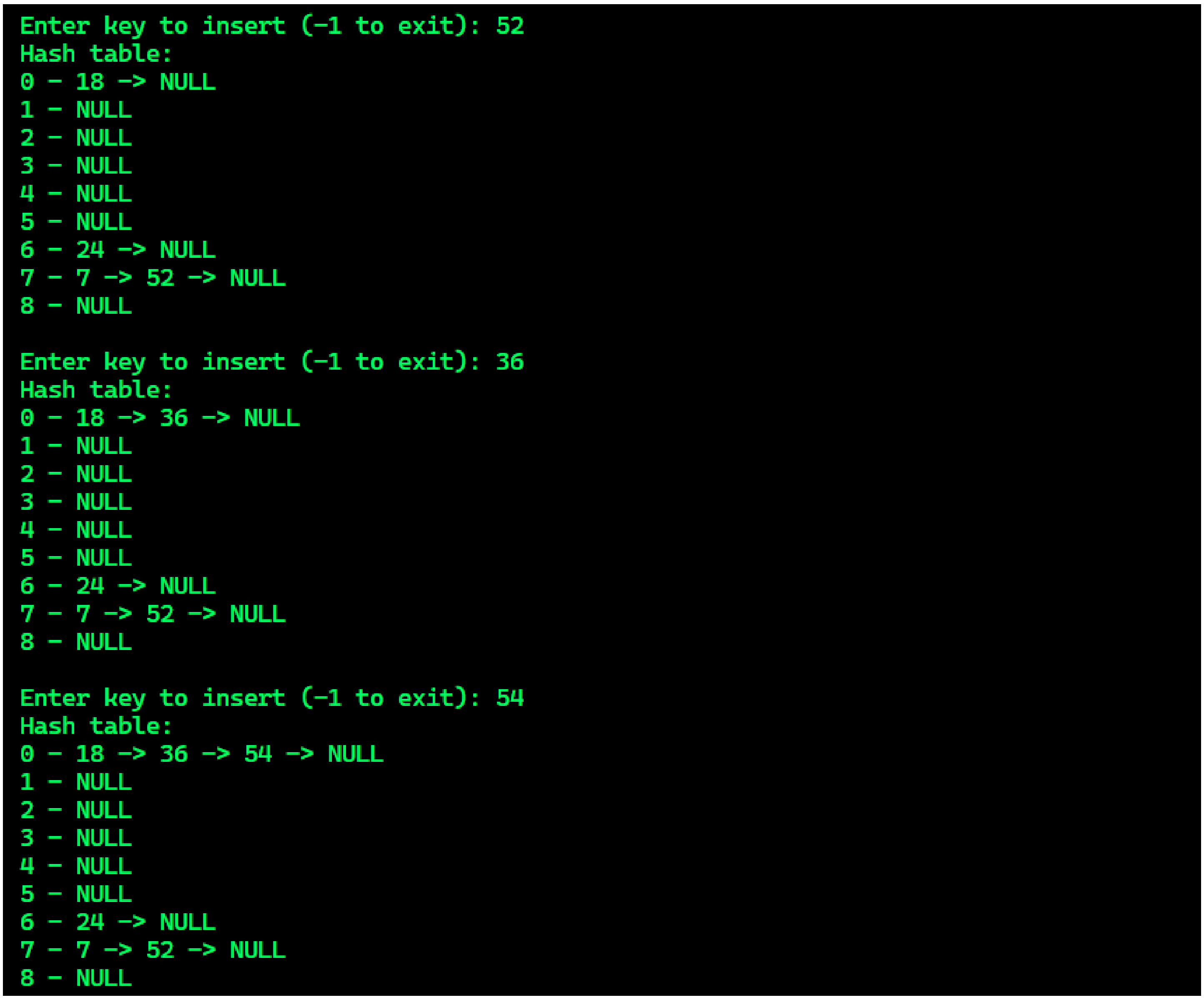
```
Enter hash table size: 10
Choose collision resolution method -
1. Linear Probing, 2. Quadratic Probing, 3. Chaining: 1

Enter key to insert (-1 to exit): 72
Hash table: - - 72 - - - - - - -
Enter key to insert (-1 to exit): 27
Hash table: - - 72 - - - - 27 - -
Enter key to insert (-1 to exit): 36
Hash table: - - 72 - - - 36 27 - -
Enter key to insert (-1 to exit): 24
Hash table: - - 72 - 24 - 36 27 - -
Enter key to insert (-1 to exit): 63
Hash table: - - 72 63 24 - 36 27 - -
Enter key to insert (-1 to exit): 81
Hash table: - 81 72 63 24 - 36 27 - -
Enter key to insert (-1 to exit): 92
Hash table: - 81 72 63 24 92 36 27 - -
Enter key to insert (-1 to exit): 101
Hash table: - 81 72 63 24 92 36 27 101 -
Enter key to insert (-1 to exit): -1
```

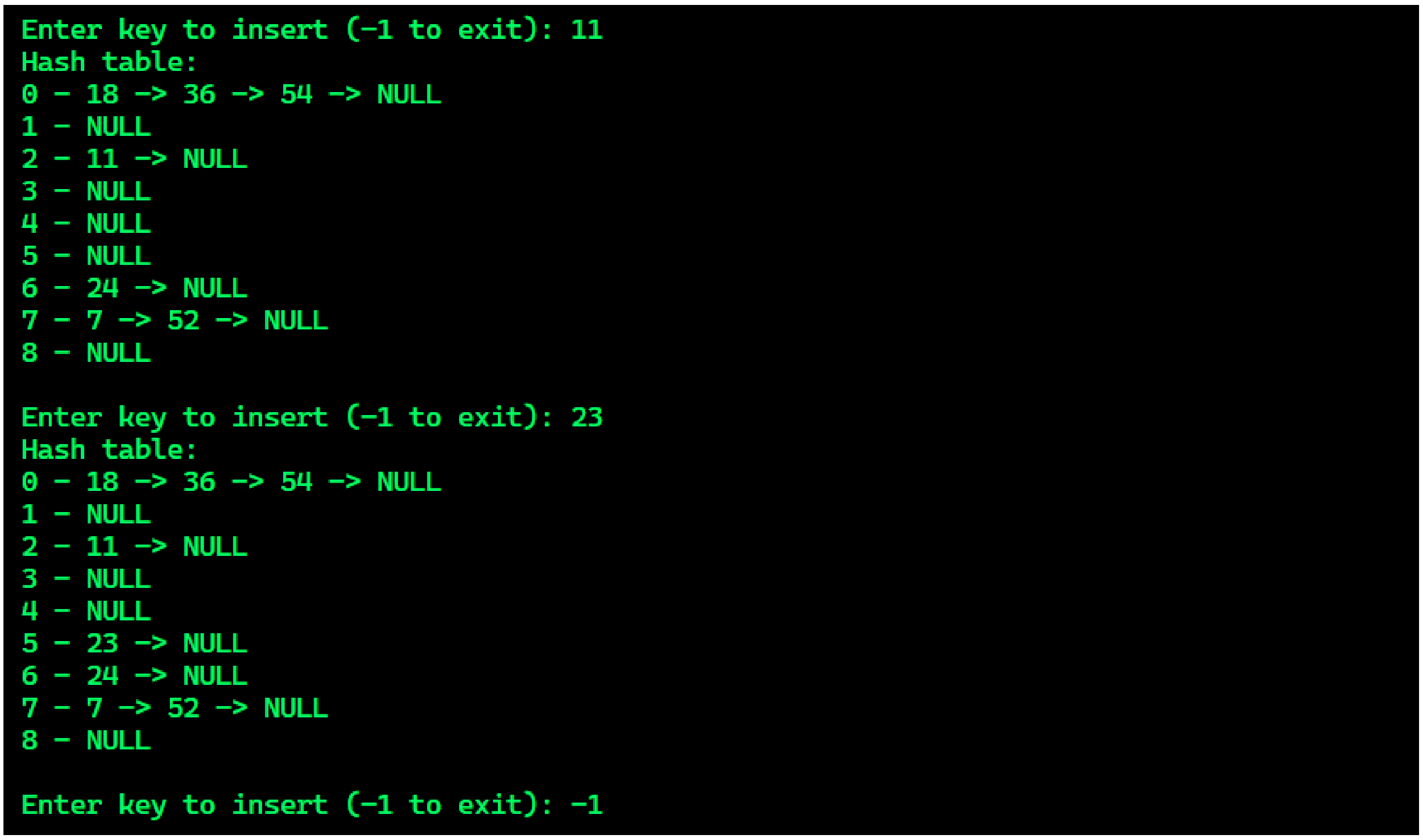


```
Enter hash table size: 10
Choose collision resolution method -
1. Linear Probing, 2. Quadratic Probing, 3. Chaining: 2
Enter values of c1 and c2: 1 3

Enter key to insert (-1 to exit): 72
Hash table: - - 72 - - - - - - -
Enter key to insert (-1 to exit): 27
Hash table: - - 72 - - - - 27 - -
Enter key to insert (-1 to exit): 36
Hash table: - - 72 - - - 36 27 - -
Enter key to insert (-1 to exit): 24
Hash table: - - 72 - 24 - 36 27 - -
Enter key to insert (-1 to exit): 63
Hash table: - - 72 63 24 - 36 27 - -
Enter key to insert (-1 to exit): 81
Hash table: - 81 72 63 24 - 36 27 - -
Enter key to insert (-1 to exit): 101
Hash table: - 81 72 63 24 101 36 27 - -
Enter key to insert (-1 to exit): -1
```



```
Enter hash table size: 9
Choose collision resolution method -
1. Linear Probing, 2. Quadratic Probing, 3. Chaining: 3

Enter key to insert (-1 to exit): 7
Hash table:
0 - NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - NULL
7 - 7 -> NULL
8 - NULL

Enter key to insert (-1 to exit): 24
Hash table:
0 - NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> NULL
8 - NULL

Enter key to insert (-1 to exit): 18
Hash table:
0 - 18 -> NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> NULL
8 - NULL
```


```
Enter key to insert (-1 to exit): 52
Hash table:
0 - 18 -> NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> 52 -> NULL
8 - NULL

Enter key to insert (-1 to exit): 36
Hash table:
0 - 18 -> 36 -> NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> 52 -> NULL
8 - NULL

Enter key to insert (-1 to exit): 54
Hash table:
0 - 18 -> 36 -> 54 -> NULL
1 - NULL
2 - NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> 52 -> NULL
8 - NULL
```



```
Enter key to insert (-1 to exit): 11
Hash table:
0 - 18 -> 36 -> 54 -> NULL
1 - NULL
2 - 11 -> NULL
3 - NULL
4 - NULL
5 - NULL
6 - 24 -> NULL
7 - 7 -> 52 -> NULL
8 - NULL

Enter key to insert (-1 to exit): 23
Hash table:
0 - 18 -> 36 -> 54 -> NULL
1 - NULL
2 - 11 -> NULL
3 - NULL
4 - NULL
5 - 23 -> NULL
6 - 24 -> NULL
7 - 7 -> 52 -> NULL
8 - NULL

Enter key to insert (-1 to exit): -1
```