```
#include <stdio.h>
#include<stdlib.h>
#define MAX 10

struct Record
{
    int data;
    struct Record *link;
};

void insert(int id, struct Record *hash_table[]);
int search_element(int key, struct Record *hash_table[]);
void remove_record(int key, struct Record *hash_table[]);
void show(struct Record *hash_table[]);
int hash_function(int key);

int main()
{
    struct Record *hash_table[MAX];
    int count, key, option,id;

    for(count = 0; count <= MAX - 1; count++)
    {
        hash_table[count] = NULL;
    }
    while(1)
    {
        printf("1. Insert a Record in Hash Table\n");
        printf("2. Search for a Record\n");
        printf("3. Delete a Record\n");
        printf("4. Show Hash Table\n");
        printf("5. Quit\n");
        printf("Enter your option\n");
        scanf("%d",&option);
        switch(option)
        {
            case 1:
                printf("Enter the number:\t");
                scanf("%d", &id);
                insert(id, hash_table);
                break;
            case 2:
                printf("Enter the element to search:\t");
                scanf("%d", &key);
                count = search_element(key, hash_table);
                if(count == -1)
                {
```

```
                printf("Element Not Found\n");
            }
            else
            {
                printf("Element Found in Chain:\t%d\n", count);
            }
            break;
        case 3:
            printf("Enter the element to delete:\t");
            scanf("%d", &key);
            remove_record(key, hash_table);
            break;
        case 4:
            show(hash_table);
            break;
        case 5:
            exit(1);
        }
    }
    return 0;
}

void insert(int id, struct Record *hash_table[])
{
    int key, h;
    struct Record *temp;
    key = id;
    if(search_element(key, hash_table) != -1)
    {
        printf("Duplicate Key\n");
        return;
    }
    h = hash_function(key);
    temp = malloc(sizeof(struct Record));
    temp->data = id;
    temp->link = hash_table[h];
    hash_table[h] = temp;

}

void show(struct Record *hash_table[])
{
    int count;
    struct Record *ptr;
    for(count = 0; count < MAX; count++)
    {
        printf("\n[%3d]", count);
```

```
        if(hash_table[count] != NULL)
        {
            ptr = hash_table[count];
            while(ptr != NULL)
            {
                printf("%d \t", ptr->data);
                ptr=ptr->link;
            }
        }
    }
    printf("\n");
}

int search_element(int key, struct Record *hash_table[])
{
    int h;
    struct Record *ptr;
    h = hash_function(key);
    ptr = hash_table[h];
    while(ptr != NULL)
    {
        if(ptr->data == key)
        {
            return h;
        }
        ptr = ptr->link;
    }
    return -1;
}

void remove_record(int key, struct Record *hash_table[])
{
    int h;
    struct Record *temp, *ptr;
    h = hash_function(key);
    if(hash_table[h]==NULL)
    {
        printf("Key %d Not Found\n", key);
        return;
    }
    if(hash_table[h]->data == key)
    {
        temp = hash_table[h];
        hash_table[h] = hash_table[h]->link;
        free(temp);
        return;
    }
```

```
    ptr = hash_table[h];
    while(ptr->link != NULL)
    {
        if(ptr->link->data == key)
        {
            temp = ptr->link;
            ptr->link = temp->link;
            free(temp);
            return;
        }
        ptr = ptr->link;
    }
    printf("Key %d Not Found\n", key);
}

int hash_function(int key)
{
    return (key % MAX);
}
```