SSN COLLEGE OF ENGINEERING, KALAVAKAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SEMESTER: III
UCS-1312: DATA STRUCTURES

NAME: A. LIKHITHA VERMA
CLASS/SEC: CSE-B
REG. NO.: 185001084

Exercise 13: Implementation of hash function

1. A. Store the following numbers in 5 buckets using any hash function (use separate chaining to avoid collision)
35, 26, 12, 24, 43, 38, 37, 41, 22, 11, 15
B. Search for an element in the hash table.
C. Delete 38 from hash table.
D. Display hash table after each operation.

2.Store the strings {“abcdef”, “bcdefa”, “cdefab” , “defabc” } using the following hash function.
The index for a specific string will be equal to sum of ASCII values of characters multiplied by their respective order in the string after which it is modulo with 2069 (prime number)

1.
PROGRAM:

```c
#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 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[%d]", count);
        if(hash_table[count] != NULL)
        {
            ptr = hash_table[count];
            while(ptr != NULL)
            {
                printf("\t %d", 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);
    }
    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 HashTable\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("\nEnter the number: ");
                scanf("%d",&id);
                insert(id,hash_table);
                break;
            case 2:
                printf("\nEnter the element to search: ");
                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: ");
                scanf("%d", &key);
                remove_record(key,hash_table);
                break;
            case 4:
                show(hash_table);
                break;
            case 5:
                exit(1);
            }
    }
    return 0;
}
```

OUTPUT:

```
(base) MSMLs-iMac:DS msml$ ./hashing
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 35
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
```

```
[1]
[2]
[3]
[4]
[5]  35
[6]
[7]
[8]
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 26
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]
[2]
[3]
[4]
[5]  35
[6]  26
[7]
[8]
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 12
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
```

```
5. Quit
Enter your option
4

[0]
[1]
[2]  12
[3]
[4]
[5]  35
[6]  26
[7]
[8]
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 24
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]
[2]  12
[3]
[4]  24
[5]  35
[6]  26
[7]
[8]
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1
```

```
Enter the number: 43
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]
[2]  12
[3]  43
[4]  24
[5]  35
[6]  26
[7]
[8]
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 38
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]
[2]  12
[3]  43
[4]  24
[5]  35
[6]  26
[7]
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
```

```
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 37
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]
[2]  12
[3]  43
[4]  24
[5]  35
[6]  26
[7]  37
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 41
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]  41
[2]  12
[3]  43
[4]  24
[5]  35
[6]  26
[7]  37
```

```
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 22
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]  41
[2]  22  12
[3]  43
[4]  24
[5]  35
[6]  26
[7]  37
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 11
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]  11  41
[2]  22  12
```

```
[3]  43
[4]  24
[5]  35
[6]  26
[7]  37
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1

Enter the number: 15
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]  11  41
[2]  22  12
[3]  43
[4]  24
[5]  15  35
[6]  26
[7]  37
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
2

Enter the element to search: 35
Element Found in Chain:  5
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
```

```
Enter your option
2

Enter the element to search: 96
Element Not Found
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
3
Enter the element to delete: 37
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[0]
[1]  11  41
[2]  22  12
[3]  43
[4]  24
[5]  15  35
[6]  26
[7]
[8]  38
[9]
1. Insert a Record in HashTable
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
5
```

2.PROGRAM:

```c
#include <stdio.h>
#include<stdlib.h>
#define MAX 40
#include<string.h>
struct Record
{
    char data[50];
    struct Record *link;
```

```
};
int hash_function(char key[])
{
    int i=0,sum=0;
    while(key[i]!='\0')
    {
       sum=sum+(int)key[i]*(i+1);
       i=i+1;
    }
    return (sum%2069);
}
void insert(char id[], struct Record *hash_table[])
{
    char key[50];int h;
    struct Record *temp; strcpy(key,id);
    if(search_element(key, hash_table) != -1)
        printf("Duplicate Key\n"); return;
    h = hash_function(key);
    temp = malloc(sizeof(struct Record));
    strcpy(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("%s \t", ptr->data);
                ptr=ptr->link;
            }
        }
    }
    printf("\n");
}
 int search_element(char key[50], struct Record
*hash_table[])
{
    int h;
    struct Record *ptr;
    h = hash_function(key);
    ptr = hash_table[h];
    while(ptr != NULL)
    {
        if(!strcmp(ptr->data,key))
            return h;
        ptr = ptr->link;
    }
    return -1;
}

void remove_record(char key[], struct Record
*hash_table[])
{
    int h;
    struct Record *temp, *ptr;
    h = hash_function(key);
    if(hash_table[h]==NULL)
    {
        printf("Key %s Not Found\n", key);
        return;
    }
    if(!strcmp(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(!strcmp(ptr->link->data,key))
        {
            temp = ptr->link;
            ptr->link = temp->link; free(temp);
            return;
        }
        ptr = ptr->link;
    }
    printf("Key %s Not Found\n", key);
}
int main()
{
    struct Record *hash_table[MAX]; int count, option;
    char key[50],id[50];
    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 string:\t");
                scanf("%s", id);
                insert(id, hash_table);
                break;
            case 2:
                printf("Enter the element to search:\t");
                scanf("%s", 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("%s", key);
                remove_record(key, hash_table);
                break;
            case 4:
                show(hash_table);
                break;
            case 5:
                exit(1);
        }
    }
    return 0;
}
```

OUTPUT:

```
(base) MSMLs-iMac:DS msml$ ./custom_hash
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1
```

```
Enter the string: abcdef
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1
Enter the string: bcdefa
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1
Enter the string: cdefab
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
1
Enter the string: defabc
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
4

[  0]
[  1]
[  2]
[  3]
[  4]
[  5]
[  6]
[  7]
[  8]
[  9]
[ 10]
[ 11]  defabc
[ 12]
[ 13]
[ 14]  cdefab
[ 15]
[ 16]
[ 17]
[ 18]
[ 19]
[ 20]
[ 21]
[ 22]
[ 23]  bcdefa
[ 24]
[ 25]
[ 26]
[ 27]
[ 28]
[ 29]
[ 30]
[ 31]
[ 32]
[ 33]
[ 34]
[ 35]
[ 36]
[ 37]
[ 38]  abcdef
[ 39]
1. Insert a Record in Hash Table
2. Search for a Record
3. Delete a Record
4. Show Hash Table
5. Quit
Enter your option
5
```