SSN COLLEGE OF ENGINEERING (Autonomous)

Affiliated to Anna University

DEPARTMENT OF CSE

UCS308 Data Structures Lab

Assignment 13

Hashing

Register Number : 185001131

Name : Sai Charan B

Class : CSE – B

A. FILE NAME : sepchain.h

```
#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);

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->link != NULL)
{
      printf("%d -> ", ptr->data);
```

```
      ptr=ptr->link;
 }
 printf("%d", ptr->data);
 }
 }
 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);
}
```

FILE NAME : main.c

```
#include “sepchain.h”
```

```
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("\n1. 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 : ");
 scanf("%d",&option);
 printf("\n");
 switch(option)
 {
 case 1:
 printf("Enter the number : ");
 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;
}
```

Output:

```
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 number : 35

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 number : 26

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 number : 12

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 number : 24

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 number : 43

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 number : 38

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 number : 37

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 number : 41

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 number : 22

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 number : 11

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 number : 15

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]11 -> 41
[  2]22 -> 12
```

```
[  3]43
[  4]24
[  5]15 -> 35
[  6]26
[  7]37
[  8]38
[  9]

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 : 2

Enter the element to search:    38
Element Found in Chain: 8

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 : 3

Enter the element to delete:    38

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]11 -> 41
[  2]22 -> 12
[  3]43
[  4]24
[  5]15 -> 35
[  6]26
[  7]37
[  8]
[  9]

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
```

B. FILE NAME : hash.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 50
struct Record
{
 char data[100];
 struct Record *link;
};

void insert(char *id, struct Record *hash_table[]);
int search_element(char *key, struct Record *hash_table[]);
```

```
void remove_record(char *key, struct Record *hash_table[]);
void show(struct Record *hash_table[]);
int hash_function(char *key);

void insert(char *id, struct Record *hash_table[])
{
 char key[20];
 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->link != NULL)
```

```
        {
             printf("%s -> ", ptr->data);
             ptr=ptr->link;
        }
        printf("%s", ptr->data);
 }
 }
 printf("\n");
}

int search_element(char key[], 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 hash_function(char *key)
{
      int hashval = 0;
      for(int i = 0; i<strlen(key); i++)
      {
            hashval += (key[i])*(i + 1);
      }
```

```
      return hashval%2069;
}
```

FILE NAME : main.c

```
#include “hash.h”

int main()
{
 struct Record *hash_table[MAX];
 int count;
 char key[20];
 int option;
 char id[20];

 for(count = 0; count <= MAX - 1; count++)
 {
 hash_table[count] = NULL;
 }
 while(1)
 {
 printf("\n1. 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 : ");
 scanf("%d",&option);
 printf("\n");
 switch(option)
 {
 case 1:
 printf("Enter the string : ");
```

```
 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

```
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 : bcdefa

Enter the string :
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]

[40]

[41]

[42]

[43]

[44]

[45]

[46]

[47]

[48]

[49]

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
```