```
//S Harshini-185001058
//1
#include<stdlib.h>
#include<stdio.h>

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

struct hash_table
{
        int s;
        struct node* list[100];
};

#define MAX 10
struct hash_table* initialize(struct hash_table *H, int size)
{
        H=(struct hash_table *)malloc(sizeof(struct hash_table));
        int i;
        H->s=size;
        for(i=0;i<H->s;i++)
        {
                H->list[i]=(struct node *)malloc(sizeof(struct node));
                H->list[i]->next=NULL;
        }

        return H;
}


void insert(struct hash_table *H, int x)
{
        int h;
        struct node* temp;
        h=x%10;
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=x;

        temp->next=H->list[h]->next;
        H->list[h]->next=temp;
```

```
        printf("\nNode Inserted %d ",temp->data);
}

void find(struct hash_table *H, int key)
{
        struct node *temp;
        int h,flag=0,count=0;
        h=key%10;
        temp=H->list[h]->next;
        while(temp!=NULL)
        {
                count++;
                if(temp->data==key)
                {
                        printf("\nData = %d",temp->data);
                        printf("\nIndex = %d",h);
                        printf("\nPosition = %d",count);
                        flag=1;
                        break;
                }
                temp=temp->next;
        }
        if(flag==0)
        {
                printf("Not Found\n");
        }
}

void display(struct hash_table *H)
{
        int i;

        struct node *temp;
        for(i=0;i<H->s;i++)
        {
                printf("[%d]-> ",i);
                temp=H->list[i]->next;
                while(temp!=0)
                {
                        if(temp->next!=NULL)
                                printf(" %d->",temp->data);
                        else
                                printf(" %d",temp->data);
```

```
                        temp=temp->next;
                }
                printf("\n");
        }
}

void remove_record(struct hash_table *H,int key)
{
 int h;
 struct node *temp ,*ptr;

 h = hash_function(key);
 if(H->list[h]==NULL)
 {
 printf("Key %d Not Found\n", key);
 return;
 }
 if(H->list[h]->data == key)
 {
 temp = H->list[h];
 H->list[h] =H->list[h]->next;
 free(temp);
 return;
 }

 ptr=H->list[h];

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

int hash_function(int key)
{
```

```
 return (key % MAX);
}
void main()
{
	struct hash_table *H;
	H=initialize(H,10);

	int ch=1;
	printf("enter the provision for hashing table::\n");

	while(ch!=5)
	{
		printf("\nEnter\n1)Insert\n2)Search\n3)Delete\n4)Display\n5)Exit\n");
		scanf("%d",&ch);

		switch(ch)
		{
			case 1:
			{
				int temp;
				printf("\nEnter a number : ");
				scanf("%d",&temp);
				insert(H,temp);
				break;
			}
			case 2:
			{
				int temp;
				printf("\nEnter a number to be searched:");
				scanf("%d",&temp);
				find(H,temp);
				break;
			}
			case 3:
			{
				int temp;
				printf("\n enter no to be deleted:");
				scanf("%d",&temp);
				remove_record(H,temp);
				break;
			}
			case 4:
			{
```

```
                display(H);
                break;
            }
            case 5:
            {
                printf("END");
                break;
            }

            default:
            {
                printf("Invalid Input try again:\n");
                break;
            }
        }
    }

}
/*SAMPLE INPUT/OUTPUT
enter the provision for hashing table::

Enter
1)Insert
2)Search
3)Delete
4)Display
5)Exit
1

Enter a number : 34

Node Inserted 34
Enter
1)Insert
2)Search
3)Delete
4)Display
5)Exit
1

Enter a number : 44

Node Inserted 44
```

```
Enter
1)Insert
2)Search
3)Delete
4)Display
5)Exit
1

Enter a number : 56

Node Inserted 56
Enter
1)Insert
2)Search
3)Delete
4)Display
5)Exit
3

 enter no to be deleted:56

Enter
1)Insert
2)Search
3)Delete
4)Display
5)Exit
4
[0]->
[1]->
[2]->
[3]->
[4]->  44-> 34
[5]->
[6]->
[7]->
[8]->
[9]->

Enter
1)Insert
2)Search
3)Delete
```

4)Display
5)Exit
5
*/

//2
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int hash(char str[]){
	int i,tot=0;
	for(i=0;str[i];i++){
		tot+=(((int)str[i])*(i+1));
	}
	int hash=tot%2069;
	return(hash);
}

char ** create(){
	char **arr;
	arr=(char **)malloc(sizeof(char *)*2069);
	int i;
	for(i=0;i<2069;i++){
		arr[i]=(char *)malloc(sizeof(char )*100);
	}
	return arr;
}

void insert(char **arr,char str[]){
	int index=hash(str);
	printf("\nIndex:%d",index);
	strcpy(arr[index],str);
}


void main(){
	char str[100],**arr;
	arr=create();
	printf("\n\nEnter string to enter:");
	scanf("%s",str);
	do{
		insert(arr,str);
```

```
                printf("\n\nEnter string to enter or END to exit:");
                scanf("%s",str);
        }while(strcmpi(str,"END"));
}

/*SAMPLE INPUT/OUTPUT

Enter string to enter:abcdef

Index:38

Enter string to enter or END to exit:bcdefa

Index:23

Enter string to enter or END to exit:cdefab

Index:14

Enter string to enter or END to exit:defabc

Index:11

Enter string to enter or END to exit:EXIT
*/
```