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.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#define TABLE_SIZE 5
struct node
{
            int data;
            struct node *next;
};
struct node *head[TABLE_SIZE]={NULL},*c,*prev;
void insert()
{
   int i,key,n;
   printf("Enter the number of terms");
   scanf("%d",&n);
   for(int j=0;j<n;j++)
   {
            printf("\nenter a value to insert into hash table\n");
            scanf("%d",&key);
            i=key%TABLE_SIZE;
            struct node * newnode=(struct node *)malloc(sizeof(struct node));
            newnode->data=key;
            newnode->next = NULL;
            if(head[i] == NULL)
                  head[i] = newnode;
            else
            {
                  c=head[i];
                  while(c->next != NULL)
                  {
                    c=c->next;
                  }
                  c->next=newnode;
            }
    }
}
```

```
void search()
{
   int key,index;
   printf("\nenter the element to be searched\n");
   scanf("%d",&key);
   index=key%TABLE_SIZE;
   if(head[index] == NULL)
      printf("\n Search element not found\n");
   else
   {
      for(c=head[index];c!=NULL;c=c->next)
      {
         if(c->data == key)
            {
               printf("search element found\n");
               break;
            }
      }
      if(c==NULL)
         printf("\n Search element not found\n");

   }
}
void display()
{
   int i;
   for(i=0;i<TABLE_SIZE;i++)
        {
        printf("\nentries at index %d\n",i);
           if(head[i] == NULL)
           {
           printf("No Hash Entry");
           return;
           }
           else
           {
                      for(c=head[i];c!=NULL;c=c->next)
                      printf("%d->",c->data);
           }
        }

}

void delete()
{
   int key,index;
   printf("\nenter the element to be deleted\n");
   scanf("%d",&key);
   index=key%TABLE_SIZE;

   struct node* temp = head[index], *prev;

 if(head[index] == NULL)
       printf("\n element not found\n");
   else
   {
      for(c=head[index];c!=NULL;c=c->next)
      {
         if(c->data == key)
            {
              if (temp != NULL && temp->data == key)
                     {
                           *head = temp->next;
                           free(temp);
                           return;
                     }

                     while (temp != NULL && temp->data != key)
                     {
                           prev = temp;
                           temp = temp->next;
                     }

                     if (temp == NULL) return;

                     prev->next = temp->next;

                     free(temp);
                     printf("\n Element deleted\n");
               break;
            }
      }
      if(c==NULL)
         printf("\n element not found\n");

   }

}
void main()
{
   int opt,key,i;
   while(1)
   {
      printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.delete \t5.Exit \n");
      scanf("%d",&opt);
      switch(opt)
```

```
        {
          case 1:
             insert();
             break;
          case 2:
             display();
             break;
          case 3:
             search();
             break;
          case 4:
             delete();
             break;
          case 5:exit(0);

        }
    }
}
```

OUTPUT:

```
Press 1. Insert   2. Display      3. Search        4.delete          5.Exit
1
Enter the number of terms
11

enter a value to insert into hash table
35

enter a value to insert into hash table
26

enter a value to insert into hash table
12

enter a value to insert into hash table
24

enter a value to insert into hash table
43

enter a value to insert into hash table
```

```
38

enter a value to insert into hash table
37

enter a value to insert into hash table
41

enter a value to insert into hash table
22

enter a value to insert into hash table
11

enter a value to insert into hash table
15

Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
2

entries at index 0
35->15->
entries at index 1
26->41->11->
entries at index 2
12->37->22->
entries at index 3
43->38->
entries at index 4
24->
Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
3

enter the element to be searched
11
search element found

Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
3

enter the element to be searched
10

 Search element not found

Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
4
```

```
enter the element to be deleted
38

 Element deleted

Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
2

entries at index 0
35->15->
entries at index 1
26->41->11->
entries at index 2
12->37->22->
entries at index 3
43->
entries at index 4
24->
Press 1. Insert  2. Display      3. Search        4.delete          5.Exit
5
```

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)

CODE:

```
#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 the String (Type END to Exit): \n");
                scanf("%s",str);
        }while(strcmp(str,"END"));
}
```

OUTPUT:

```
Enter string to enter:abcdef

Index:38

Enter the String (Type END to Exit):
bcdefa

Index:23

Enter the String (Type END to Exit):
cdefab

Index:14

Enter the String (Type END to Exit):
defabc

Index:11

Enter the String (Type END to Exit):
END
```