```c
/*
12. Given a File of N employee records with a set K of Keys(4-digit) which uniquely
determine the records in file F. Assume that file F is maintained in memory by a
Hash Table(HT) of m memory locations with L as the set of memory addresses(2-digit)
of locations in HT. Let the keys in K and addresses in L are Integers.
Design and develop a Program in C that uses Hash function H: K -> L as
H(K)=K mod m (remainder method) and implement hashing technique to map
a given key K to the address space L.
Resolve the collision (if any) using linear probing.
*/

#include <stdio.h>
#include <math.h>

#define MAX 10                          //Maximum size of hash table is 10 [0 to 9]]

/*
The program has 3 parts: 1-main function, 2-Linear probing, 3- Display
First we initialise the entire hash table with -11111.
Now -11111 here indicates empty places in hash table.
*/

void main()
{
    int a[MAX];
    int num;
    int i;
    int ch;

    for (i=0;i<MAX;i++)
        a[i] = -11111;              //initialize entire HT with -11111 entries

    while(1)
        {
            printf("\n ***Collision handling by Linear Probing***\n");

            printf("1 - Insert into Hash table\n");
            printf("2 - Display Hash table\n");
            printf("3 - Exit\n");

            printf("Enter your Choice :");
            scanf("%d",&ch);

            switch (ch)
            {
                case 1: linearprob(a,num);  //Function call to linearprob() function
                        break;

                case 2: display(a);         //Function call to display() function
                        break;

                case 3: return;
                default: printf("Invalid Choice\n");
            }
        }
}

/*
num is 4 digit key. Not to be confused with 'key'.
key is the index value of where the number must be stored.

flag is used to specify whether key is entered or not.
By default, flag=0 specifying key is not entered.

If the collision is detected,we check for next available empty location.
If collision is detected & also space is full,we say Hash table FULL.
```

```c
*/

void linearprob(int a[MAX], int num)          //linearprob() function
{
    int flag;
    int i;
    int key;
    int count;
    char ans;
    do
        {
            flag=0;               //Specifies initially no number is entered
            count=0;             //keeps the numbers put into hash table

            printf("Enter 4 digit Key : ");
            scanf("%4d", &num);                    // reads 4-digit a number

            key=num%10;             //generates single digit key for given number

            if(a[key]== -11111)         // check for empty entry in Hash table
                a[key] = num;     //if yes then add
            else                    // if entry exits then its must avoid collision
                {
                    printf(" Collision Detected...!!!\n");

                    i=0;
                    while(i<MAX)   // check for next available empty location in HT
                        {
                            if (a[i]!=-11111)
                            count++;     //increment locations that are filled up
                            i++;
                        }                                   // end of while

                    if(count == MAX) // if HT is full then display HT and return
                        {
                            printf("\n Hash table is full \n");
                            display(a);                              // Display HT
                            return;
                        }

                    printf("Collision avoided successfully using LINEAR PROBING\n");
/*
If there is empty space after a place where collision is detected in HT then make a
entry of the num in that place in the HT.
This is represented by i=key+1 and if a[i]==-1111.
i=key+1 means start from one place after where the collision was detected.
if a[i]==-1111 means if that place is empty.
*/
                    for(i=key+1; i<MAX; i++)
                        if(a[i] == -11111)
                        {
                            a[i] = num;         //insert the num in HT
                            flag =1;          //Mark the location as occupied
                            break;
                        }
                        i=0;
/*
Check for empty space before key in HT then make a entry in HT.
If there is empty space after a place where collision is detected in HT then make a
entry of the num in that place in the HT.
This is represented by i<key and flag==0.
i<key represents the look for an empty space prior to where collision occured.
and check if its empty (flag==0)
*/
                        while((i<key) && (flag==0))
                            {
```

```
                                if(a[i] == -11111)        //if location is empty
                                {
                                    a[i] = num;        //insert the num in HT
                                    flag=1;      //Mark the location as occupied
                                    break;
                                }
                                i++;
                            }
                }

        } while(ans== 'y'||ans == 'Y');         // end of do-while statement

} // end of if statement

void display(int a[MAX])                         // display() function
{
    int i;
    printf("The hash table is \n Key \t Value\n");

    for(i=0; i<MAX; i++)
    printf(" %d\t %d\n", i, a[i]);
}
```

OUTPUT:

```
 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       -11111
 1       -11111
 2       -11111
 3       -11111
 4       -11111
 5       -11111
 6       -11111
 7       -11111
 8       -11111
 9       -11111

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 1234

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 2346

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 7777

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
```

```
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       -11111
 1       -11111
 2       -11111
 3       -11111
 4       1234
 5       -11111
 6       2346
 7       7777
 8       -11111
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 3456
 Collision Detected...!!!
Collision avoided successfully using LINEAR PROBING

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       -11111
 1       -11111
 2       -11111
 3       -11111
 4       1234
 5       -11111
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 1244
 Collision Detected...!!!
```

```
Collision avoided successfully using LINEAR PROBING

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       -11111
 1       -11111
 2       -11111
 3       -11111
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 5555
 Collision Detected...!!!
Collision avoided successfully using LINEAR PROBING

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       5555
 1       -11111
 2       -11111
 3       -11111
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
```

```
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 3457
 Collision Detected...!!!
Collision avoided successfully using LINEAR PROBING

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       5555
 1       3457
 2       -11111
 3       -11111
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 2222

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       5555
 1       3457
 2       2222
 3       -11111
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
```

```
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 3333

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :2
The hash table is
 Key     Value
 0       5555
 1       3457
 2       2222
 3       3333
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
Enter your Choice :1
Enter 4 digit Key : 5666
 Collision Detected...!!!

 Hash table is full
The hash table is
 Key     Value
 0       5555
 1       3457
 2       2222
 3       3333
 4       1234
 5       1244
 6       2346
 7       7777
 8       3456
 9       9999

 ***Collision handling by Linear Probing***
1 - Insert into Hash table
2 - Display Hash table
3 - Exit
```

Enter your Choice :3