**Exercise 14:** **Implementation of Hash Table**

**adt.h:**

#include<stdio.h>

struct hash{

int arr[1000];

int size;

};

void init(struct hash\* h,int n);

void insert(struct hash \*h,int x);

void display(struct hash \*h);

void find(struct hash \*h,int x);

**impl.h:**

#include "adt.h"

void init(struct hash\* h,int n)

{

h->size = n;

for(int i=0;i<n;i++)

{

h->arr[i]=0;

}

}

void insert(struct hash \*h,int x)

{

int probes=0;

int si = x%(h->size);

int idx=si;

do{

if(!h->arr[si]){

h->arr[si]=x;

break;

}

else{

si = (si+1)%(h->size);

probes++;

}

}while(idx!=si);

//printf("probes:%d\n",probes);

printf("probes for element %d : %d\n",x,probes);

}

void display(struct hash \*h)

{

for(int i=0;i<h->size;i++)

{

if(h->arr[i]!=0)

printf("%d->%d\n",i,h->arr[i]);

else

printf("%d->\n",i);

}

}

void find(struct hash \*h,int x)

{

int probes=0;

int si = x%(h->size);

int idx=si;

do{

if(h->arr[si]==x){

printf("Element %d found at index : %d\t",x,si);

break;

}

else{

si = (si+1)%(h->size);

probes++;

}

}while(idx!=si);

printf("probes : %d\n",probes);

}

int find1(struct hash \*h,int x)

{

int probes=0;

int si = x%(h->size);

int idx=si;

do{

if(h->arr[si]==x){

//printf("Element %d found at index:%d\n",x,si);

return x;

//break;

}

else{

si = (si+1)%(h->size);

probes++;

}

}while(idx!=si);

//printf("probes:%d\n",probes);

return 0;

}

**appl.c:**

#include "impl.h"

#include<stdlib.h>

void insert2(struct hash \*h,int x);

void insert3(struct hash \*h,int x);

void range(struct hash \*h,int m,int n);

int main()

{

struct hash \*h = (struct hash \*)malloc(sizeof(struct hash));

int ch;

printf("\nMenu:\n1.Insert(Linear)\n2.Display the array\n3.Find an element\n4.Quadratic probing\n5.Double Hashing\n6.Find elements in a given range\n 7..Exit\n");

do

{

printf("\nChoice: ");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter the number of elements: ");

int n,ele;//,A[20];//={49,29,2,5,11,8,63,72,83};

scanf("%d",&n);

init(h,10);

for(int i=0;i<n;i++)

{

printf("Enter the element: ");

scanf("%d",&ele);

insert(h,ele);

}

//sizeof(A)/sizeof(A[0]));

break;

case 2:

display(h);

break;

case 3:

printf("\nEnter the element to be found: ");

scanf("%d",&ele);

find(h,ele);

break;

case 4:

init(h,10);

printf("\nEnter the number of elements: ");

scanf("%d",&n);

init(h,10);

for(int i=0;i<n;i++)

{

printf("Enter the element: ");

scanf("%d",&ele);

insert2(h,ele);

}

/\*insert2(h,23);

insert2(h,45);

insert2(h,69);

insert2(h,87);

insert2(h,48);

insert2(h,67);

insert2(h,54);

insert2(h,66);

insert2(h,53);\*/

display(h);

break;

case 7:printf("\nExiting...\n");

break;

case 5:

init(h,10);

printf("\nEnter the number of elements: ");

scanf("%d",&n);

init(h,10);

for(int i=0;i<n;i++)

{

printf("Enter the element: ");

scanf("%d",&ele);

insert3(h,ele);

}

/\*insert3(h,23);

insert3(h,45);

insert3(h,69);

insert3(h,87);

insert3(h,48);

insert3(h,67);

insert3(h,54);

insert3(h,66);

insert3(h,53);\*/

display(h);

break;

case 6:

printf("\nEnter the a & b [a,b]: ");

int a,b;

scanf("%d%d",&a,&b);

range(h,a,b);

break;

default:printf("\nInavlid Input!\n");

}

}while(ch!=6);

/\*printf("\nLinear probing:\n");

init(h,10);

insert(h,23);

insert(h,45);

insert(h,69);

insert(h,87);

insert(h,48);

insert(h,67);

insert(h,54);

insert(h,66);

insert(h,53);\*/

//display(h);

//find(h,69);

//range(h,40,60);

/\*printf("\nQuadratic probing:\n");

init(h,10);

insert2(h,23);

insert2(h,45);

insert2(h,69);

insert2(h,87);

insert2(h,48);

insert2(h,67);

insert2(h,54);

insert2(h,66);

insert2(h,53);

display(h);

find(h,69);

range(h,40,60);\*/

/\*printf("\n\n");

init(h,10);

insert3(h,23);

insert3(h,45);

insert3(h,69);

insert3(h,87);

insert3(h,48);

insert3(h,67);

insert3(h,54);

insert3(h,66);

insert3(h,53);

display(h);

find(h,69);

range(h,40,60);\*/

return 0;

}

void insert2(struct hash \*h,int x)

{

int probes=0;

int si = x%(h->size);

int idx=si;

int var=si;

do{

if(!h->arr[si]){

h->arr[si]=x;

break;

}

else{

si= (var+(++probes)\*(probes))%(h->size);

}

}while(idx!=si);

//printf("probes:%d\n",probes);

printf("probes for element %d : %d\n",x,probes);

}

void insert3(struct hash \*h,int x)

{

int probes=0;

int si = x%(h->size);

int idx=si;

int var=si;

do{

if(!h->arr[si]){

h->arr[si]=x;

break;

}

else{

si = (si+(7-x%7))%(h->size);

probes++;

}

}while(idx!=si);

printf("probes for element %d : %d\n",x,probes);

}

void range(struct hash \*h,int m,int n)

{

for(int i=m;i<n;i++)

{

int x=find1(h,i);

if(x)

{

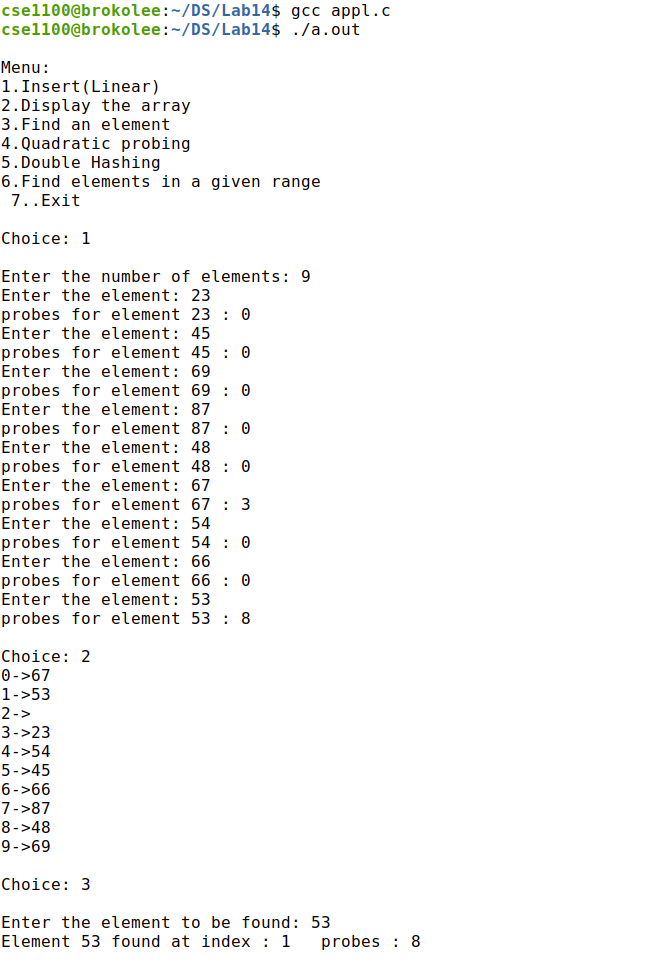
printf("Found: \t%d\n",x);

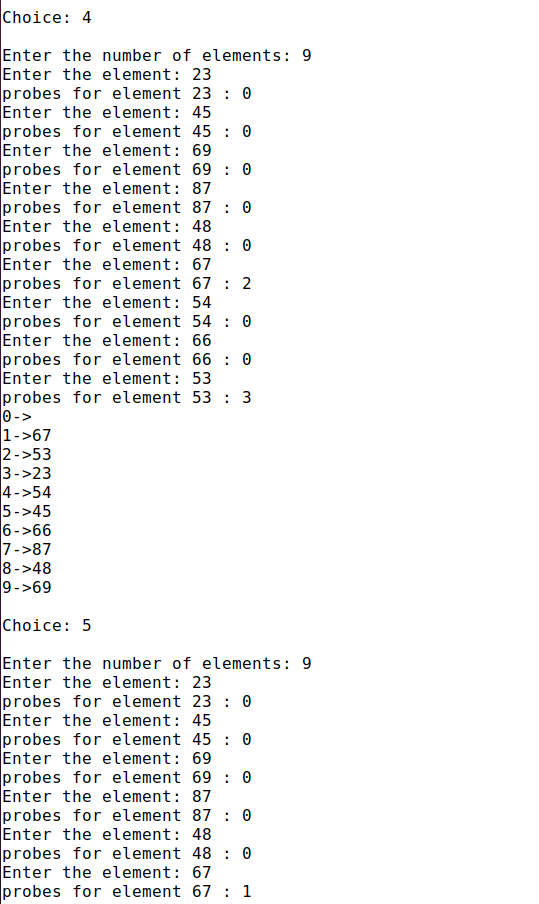
}

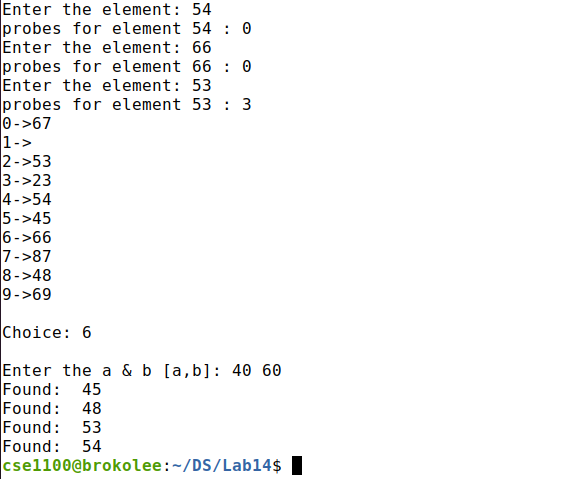
}

}

*Sample I/O:*

**



**