**PROGRAM CODE:**

#include<stdio.h>

#include<stdlib.h>

#include "SC.h"

#include "LP.h"

void main()

{

hashtable t=NULL;

LinearTable lt=NULL;

int ch1,ch2,tsize,ltsize,data;

do

{

printf("\n1.Separate Chaining\n2.Linear Probing\n3.Exit\n\nEnter choice: ");

scanf("%d",&ch1);

if(ch1==1)

{

printf("Enter the table Size: ");

scanf("%d",&tsize);

t=initializetable(tsize);

printf("\n1.To enter the values\n2.Exit\n\nEnter choice: ");

scanf("%d",&ch2);

if(ch2==1)

{

printf("Enter -1 to stop\n");

printf("\nEnter value: ");

scanf("%d",&data);

do{

insert(data,t);

//disp(t);

printf("\nEnter value: ");

scanf("%d",&data);

}while(data!=-1);

printf("\nThe hash table is\n");

display(t);

}

else if(ch2==2) {}

else

printf("Invalid choice");

}

else if(ch1==2)

{

printf("Enter the table Size: ");

scanf("%d",&ltsize);

lt=create(ltsize);

printf("\n1.To enter the values\n2.Exit\n\nEnter choice: ");

scanf("%d",&ch2);

if(ch2==1)

{

printf("Enter -1 to stop\n");

printf("\nEnter value: ");

scanf("%d",&data);

do{

insertlinear(lt,data);

//displaylinear(t);

printf("\nEnter value: ");

scanf("%d",&data);

}while(data!=-1);

printf("\nThe hash table is\n");

displaylinear(lt);

}

else if(ch2==2) {}

else

printf("Invalid choice");

}

}while(ch1!=3);

}

***LP header file***

typedef struct lineartable\* LinearTable;

struct lineartable {

int lt\_size;

int\* arr;

};

LinearTable create(int tsize)

{

LinearTable l=malloc(sizeof(struct lineartable));

l->lt\_size=tsize;

l->arr=(int\*)malloc(sizeof(int)\*tsize);

for(int i=0;i<l->lt\_size;i++)

l->arr[i]=-1;

return l;

}

void displaylinear(LinearTable l)

{

for(int i=0;i<l->lt\_size;i++)

{

printf("%d: ",i);

if(l->arr[i]!=-1)

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

else

printf("\n");

}

}

void insertlinear(LinearTable l,int data)

{

int pos,ins=0,temp;

static int cnt=0;

if(cnt==l->lt\_size)

printf("NO SPACE\n");

else

{

pos=data%l->lt\_size;

while((l->arr[pos]!=-1))

pos=(pos+1)%l->lt\_size;

l->arr[pos]=data;

cnt++;

}

}

***SC header file***

typedef struct listnode \*Node;

typedef struct listnode \*position;

typedef struct hashtbl \*hashtable;

typedef position list;

typedef int elementtype;

struct listnode

{

elementtype element;

position next;

};

struct hashtbl

{

int tablesize;

list \*thelists;

};

int nextprime(int x)

{

int i,j,c;

for(i=x;;i++)

{

c=0;

for(j=2;j<=(i/2);j++)

if(i%j==0)

c++;

if(c==0)

return i;

}

}

int hash(int key, int tablesize)

{

return key % tablesize;

}

position find(elementtype key,hashtable h)

{

position pos;

pos=h->thelists[hash(key,h->tablesize)];

return pos;

}

void display(hashtable t)

{

Node temp;

for(int i=0;i<t->tablesize;i++)

{

printf("%d ",i);

temp=t->thelists[i];

while(temp->next!=NULL)

{

temp=temp->next;

printf("-> %d ",temp->element);

}

printf("\n");

}

}

hashtable initializetable(int tablesize)

{

int i;

hashtable h = malloc(sizeof(struct hashtbl));

h->tablesize = nextprime(tablesize);

h->thelists = malloc(sizeof(list) \* h->tablesize);

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

{

h->thelists[i]=malloc(sizeof(struct listnode));

h->thelists[i]->next=NULL;

}

return h;

}

void insert(elementtype key,hashtable h)

{

position newcell;

list l;

l=find(key,h);

newcell=malloc(sizeof(struct listnode));

newcell->element=key;

newcell->next=NULL;

if(l->next==NULL)

l->next=newcell;

else

{

newcell->next=l->next;

l->next=newcell;

}

}

**OUTPUT**:











