**Implementation of hash table with overflow handling technique**

**Program**

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 5

int h[TABLE\_SIZE]={0};

void insert()

{

int key,index,i,flag=0,hkey;

printf("\nEnter a value to insert into hash table\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE;i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index] == 0)

{

h[index]=key;

break;

}

}

if(i == TABLE\_SIZE)

printf("\nElement cannot be inserted!!\n");

}

void search()

{

int key,index,i,flag=0,hkey;

printf("\nEnter search element\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE; i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index]==key)

{

printf("Value is found at index %d",index);

break;

}

}

if(i == TABLE\_SIZE)

printf("\n Value is not found\n");

}

void display()

{

int i;

printf("\nElements in the hash table are \n");

for(i=0;i< TABLE\_SIZE; i++)

printf("\nat index %d \t value = %d",i,h[i]);

}

int main()

{

int opt,i;

while(1)

{

printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:exit(0);

}

}

}

Program

#include<stdlib.h>

#include<stdio.h>

#include<limits.h>

typedef struct node{

struct node\* prev{

int value;

int size;

struct node\* next;

}node;

node\* create()

{ node\* t = malloc(sizeof(node));

printf("Enter the value : ");

scanf("%d",&(t->value));

printf("Enter the size : ");

scanf("%d",&(t->size));

t->next=NULL;

t->prev=NULL;

return t; }

node\* insert(node\* t,node\* H){

// inserting at front if(H==NULL){ H=t; }

else{ node\* p = H; while(p->next)

{ p=p->next; }

p->next=t; t->prev=p; } return H; }

node\* insertGarbage(int p,node\* H)

{ // inserting garbage node\* t= malloc(sizeof(node));

t->next=NULL;

t->prev=NULL;

t->size=p;

t->value=INT\_MIN;

if(H==NULL){ H=t;

}

else{ node\* p = H; while(p->next)

{ p=p->next; }

p->next=t; t->prev=p; }

return H; }

void display(node \* H)

{ printf("\nThe current list is : ");

node\* p = H;

if(p){

while(p->next)

{ if(p->value!=INT\_MIN)

{ printf("%d -> ",p->value);

} else

{ printf("# -> ");

} p=p->next;

}

if(p->value!=INT\_MIN)

{ printf("%d -> ",p->value); }

else{ printf("# -> "); }

} printf("\nThe corresponding sizes are : ");

p = H;

if(p)

{ while(p->next){ printf("%d -> ",p->size);

p=p->next; } printf("%d -> ",p->size);

}

printf("\n");

}

void displayGarbage(node \* HG){

printf("\nThe garbage values is : ");

node\* p = HG;

if(p){

while(p->next){

printf("%d -> ",p->size);

p=p->next;

}

printf("%d -> ",p->size);

}

}

node\* delete(node\* H,node\* HG){

int pos;

printf("Enter the pos to delete : ");

scanf("%d",&pos);

node\* p = H;

for(int i =2;i<=pos;i++){

p=p->next;

}

if(p->value!=INT\_MIN){

p->value=INT\_MIN;

HG= insertGarbage(p->size,HG);

}

display(H);

return HG;

}

void garbageCollection(node\* H,node\* HG){

node \* p =H;

int garbage = 0;

if(p){

while(p->next){

if(p->value==INT\_MIN){

garbage += p->size;

}

p=p->next;

}

if(p->value==INT\_MIN){

garbage += p->size;

}

}

printf("The amount of garbage collected is : %d",garbage);

displayGarbage(HG);

}

int main(){

node\* HL = NULL;

node\* HG = NULL;

int n,pos;

while (1){

printf("\n============================");

printf("\n1.Display\n2.Insert\n3.Delete\n4.Collect Garbage\n5.Exit");

printf("\n============================");

printf("\nEnter Choice : ");

fflush(stdin);

scanf("%d", &n);

node\* t;

switch (n) {

case 1:

display(HL);

break;

case 2:

t=create();

HL=insert(t,HL);

display(HL);

break;

case 3:

HG = delete(HL,HG);

break;

case 4:

garbageCollection(HL,HG);

break;

case 5:

exit(0);

default :

printf("Wrong option selected");

}

}

}