/\* Ass 8 – Hash Table – II

Set A - Implement hash table using singly linked lists. Write a menu driven program to perform operations on hash table (insert, search, delete, display). Select appropriate hashing function. In case of collision, use separate chaining \*/

#include <stdio.h>

#include <stdlib.h>

#define TABLE\_SIZE 10

struct node

{

int data;

struct node \*next;

};

struct node \*head[TABLE\_SIZE]={NULL},\*c;

void insert()

{

int i,key;

printf("Enter a value to insert into hash table: ");

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("Enter the element to be searched: ");

scanf("%d",&key);

index=key%TABLE\_SIZE;

if(head[index] == NULL)

printf("Search element not found!");

else

{

for(c=head[index];c!=NULL;c=c->next)

{

if(c->data == key)

{

printf("Search element found!");

break;

}

}

if(c==NULL)

printf("Search element not found!");

}

}

void display()

{

int i;

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

{

printf("\nEntries at index %d: ",i);

if(head[i] == NULL)

{

printf("No Hash Entry!");

//return;

}

else

{

for(c=head[i];c!=NULL;c=c->next)

printf("%d->",c->data);

}

}

}

int main()

{

int opt,key,i;

while(1)

{

printf("\n1.Insert\n2.Display\n3.Search\n4.Exit\nEnter choice:");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

break;

case 4:exit(0);

default:

printf("Enter Valid Choice!");

}

}

}