ASSIGNMENT

***LINK LIST IMPLEMENTATION***

***WAQAS ASHIQ.***

***BCS-F11-201.***

***SECTION C.***

***SUBMITTED TO : SIR DR.GHULAM RASOOL.***

**Code:**

#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

}node;

void insert(node \*pointer, int data)

{

/\* Iterate through the list till we encounter the last node.\*/

while(pointer->next!=NULL)

{

pointer = pointer -> next;

}

/\* Allocate memory for the new node and put data in it.\*/

pointer->next = (node \*)malloc(sizeof(node));

pointer = pointer->next;

pointer->data = data;

pointer->next = NULL;

}

int find(node \*pointer, int key)

{

pointer = pointer -> next; //First node is dummy node.

/\* Iterate through the entire linked list and search for the key. \*/

while(pointer!=NULL)

{

if(pointer->data == key) //key is found.

{

return 1;

}

pointer = pointer -> next;//Search in the next node.

}

/\*Key is not found \*/

return 0;

}

void del(node \*pointer, int data)

{

/\* Go to the node for which the node next has to be deleted \*/

while(pointer->next!=NULL && (pointer->next)->data != data)

{

pointer = pointer -> next;

}

if(pointer->next==NULL)

{

printf("Element %d is not present in the list\n",data);

return;

}

/\* Now pointer points to a node and the node next to it has to be removed \*/

node \*temp;

temp = pointer -> next;

/\*temp points to the node which has to be removed\*/

pointer->next = temp->next;

/\*We removed the node which is next to the pointer (which is also temp) \*/

free(temp);

/\* Beacuse we deleted the node, we no longer require the memory used for it .

free() will deallocate the memory.

\*/

return;

}

void print(node \*pointer)

{

if(pointer==NULL)

{

return;

}

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

print(pointer->next);

}

int main()

{

/\* start always points to the first node of the linked list.

temp is used to point to the last node of the linked list.\*/

node \*start,\*temp;

start = (node \*)malloc(sizeof(node));

temp = start;

temp -> next = NULL;

/\* Here in this code, we take the first node as a dummy node.

The first node does not contain data, but it used because to avoid handling special cases

in insert and delete functions.

\*/

int sw;

printf("1 to Insert, 2 to Delete,3 to Print,4 to Find and 0 to end: ");

scanf("%d",&sw);

while(sw!=0)

{

switch(sw){

case 1:

int ins;

printf("\nEnter the data to be inserted: ");

scanf("%d",&ins);

insert(start,ins);

printf("1 to Insert, 2 to Delete,3 to Print,4 to Find and 0 to end: ");

scanf("%d",&sw);

break;

case 2:

int dele;

printf("\nEnter the data to be deleted: ");

scanf("%d",&dele);

del(start,dele);

printf("1 to Insert, 2 to Delete,3 to Print,4 to Find and 0 to end: ");

scanf("%d",&sw);

break;

case 3:

printf("\nThe list is ");

print(start->next);

printf("\n");

printf("1 to Insert, 2 to Delete,3 to Print,4 to Find and 0 to end: ");

scanf("%d",&sw);

break;

case 4:

int ser;

printf("Enter the element you want to search: ");

scanf("%d",&ser);

int status = find(start,ser);

if(status)

{

printf("\nElement Found\n");

}

else

{

printf("\nElement Not Found\n");

}

printf("1 to Insert, 2 to Delete,3 to Print,4 to Find and 0 to end: ");

scanf("%d",&sw);

break;

}

}

}

OutPut:

