#include <stdio.h>

#include <stdlib.h>

void insert();

void delet();

void display();

struct node

{

int data;

struct node \*link;

};

struct node \*first=NULL,\*last=NULL,\*next,\*prev,\*cur;

int nodecount=0;

void insert()

{

int pos,c=1;

cur=(struct node\*)malloc(sizeof(struct node));

// printf("\nENTER THE DATA: ");

// scanf("%d",&cur->data);

// cur->link=NULL;

if(first==NULL)

{ printf("\nENTER THE DATA: ");

scanf("%d",&cur->data);

cur->link=NULL;

first=cur;

nodecount++;

}

else

{

printf("number of nodes in the list =%d \n",nodecount);

printf("\nENTER THE POSITION: 1->for front & any other value for any other position");

scanf("%d",&pos);

if(pos>nodecount+1)

{

printf("\n Enter valid position \n");

}

else

{ printf("\nENTER THE DATA: ");

scanf("%d",&cur->data);

if((pos==1) &&(first!=NULL))

{

cur->link = first;

first=cur;

}

else

{

next=first;

while(c<pos)

{

prev=next;

next=prev->link;

c++;

}

if(prev==NULL)

{

printf("\nINVALID POSITION\n");

}

else

{

cur->link=prev->link;

prev->link=cur;

}

}

nodecount++;

}

}

}

void delet()

{

int pos,c=1;

printf("\nENTER THE POSITION : ");

scanf("%d",&pos);

if(pos>nodecount)

{

printf("\n Enter valid position \n");

}

else

{

if(first==NULL)

{

printf("\nLIST IS EMPTY\n");

}

else if(pos==1 && first->link==NULL)

{

printf("\n DELETED ELEMENT IS %d\n",first->data);

free(first);

first=NULL;

}

else if(pos==1 && first->link!=NULL)

{

cur=first;

first=first->link;

cur->link=NULL;

printf("\n DELETED ELEMENT IS %d\n",cur->data);

free(cur);

}

else

{

next=first;

while(c<pos)

{

cur=next;

next=next->link;

c++;

}

cur->link=next->link;

next->link=NULL;

if(next==NULL)

{

printf("\nINVALID POSITION\n");

}

else

{

printf("\n DELETED ELEMENT IS %d\n",next->data);

free(next);

}

}

}

}

void display()

{

cur=first;

while(cur!=NULL)

{

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

cur=cur->link;

}

}

int main()

{

int ch;

printf("\n\nSINGLY LINKED LIST");

do

{

printf("\n\n 1.INSERT\n2.DELETE\n3.EXIT");

printf("\n\nENTER YOUR CHOICE : ");

scanf("%d",&ch);

switch(ch)

{

case 1:

insert();

display();

break;

case 2:

delet();

display();

break;

case 3:

exit(0);

default:

printf("Invalid choice...");

}

}while(1);

}

1. Write a C program to split the given linked list ( length >1) into two sublists — one for the front half, and one in the back half. If the number of elements is odd, the extra element should go on the front list. For example, a list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}.
2. Given two sorted lists L1 and L2 , write a C program to compute L1∩L2 ( common elements in the two list in sorted order ) using only the basic list operations. For example, if L1= { 1, 3, 5, 7, 9, 12} and L2={ 3,7,17 } then L1∩L2 = { 3,7} .
3. Given two lists L1 and L2 , write a C program to compute L1 - L2 ( remove the elements from the list L1, which are already present in L2 ) using only the basic list operations. For example, if L1= { 1, 9, 5, 7,1 9, 12} and L2={ 3,7,12 } then L1-L2 = {1, 9, 5,19} .
4. Write a C program for ‘RemoveDuplicates()’ function which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.