Experiment 1

Experiment 1:A data structure needs to be implemented in such a way that we have the references i.e. the addresses of the values. None of the addresses are in continuous memory block. Each time a new value needs to be stored, we need to allocate memory. Write a program to implement the following:

* + - 1. Addition of a new value at a given position
      2. Print all the values in the list
      3. Delete a given value from a a location

CO Attained: CO2 and CO3

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<0){return ;}

if(pos==0){insertAtBegin(x); return ;}

Node \*p1 = head;

while(--pos){

p1 = p1->next;

}

Node \*n1 = new Node(x);

n1->next = p1->next;

p1->next = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void deleteBegin(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

head = head->next;

}

void deletValue(int x){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*prev=NULL , \*curr=head;

bool flag=true;

while(curr){

if(curr->val == x){

flag=false;

Node \*temp = curr;

prev->next = curr->next;

curr->next = prev;

// delete(curr);

// delete(temp);

}

prev = curr;

curr = curr->next;

}

if(flag){

cout<<"There exist no element which is asked to delete.\n";

}else{

cout<<"Deleted\n";

}

}

void deleteAtPos(int x, int pos){

int n = getSize();

if(pos>n || pos<=0){return ;}

if(pos==1){deleteBegin(); return ;}

Node \*p1 = head, \*prev=nullptr;

while(--pos){

prev = p1;

p1 = p1->next;

}

if(p1->val==x){

prev->next = p1->next;

free(p1);

}

}

void deleteEnd(){

if(!head){

cout<<"Underflow: No data to delete.";

return ;

}

Node \*temp = head;

while(temp->next->next){

temp = temp->next;

}

temp->next = NULL;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

Node\* getHead(){return this->head;}

int getSize(){

if(!head)return 0;

if(!head->next)return 1;

Node \*curr = head;

int ans=0;

while(curr){

curr = curr->next;

ans++;

}

return ans;

}

};

int main(){

LL l1;

l1.insertAtEnd(11); //

l1.insertAtEnd(13); //

l1.insertAtEnd(12); //

l1.insertAtEnd(19); //

l1.insertAtEnd(12); //

l1.insertAtEnd(13);

l1.insertAtPos(999,1);

l1.deleteAtPos(12,2);

l1.printList();

return 0;

}