//Single Linklist implementation

#include<iostream>

using namespace std;

class node {

public:

int data;

node \*next;

};

node \*head= new node();

node \*curr= new node();

int length=0;

void GoToHead() { // set curr pointer to head node;

curr= head;

}

void insertNodeAtEnd(int val) { // This function will insert new node at the end.

GoToHead();

node \*t= new node();

while(curr->next!=NULL)

curr= curr->next;

t->data= val;

t->next= NULL;

curr->next= t;

length++;

}

void AddNodeBeforeHead( int val) { // This function will insert new node as a head.

GoToHead();

node \*t= new node();

t->data= val;

t->next= curr;

head= t;

length++;

}

void InsertAfterSpecificKey(int val, int key ) {

node \*t= new node();

GoToHead();

while (curr!=NULL) {

if (curr->data==key) {

t->data= val;

t->next= NULL;

t->next= curr->next;

curr->next= t;

length++;

break;

}

curr= curr->next;

}

}

void InsertBeforeSpecificKey(int val, int key ) {

node \*ptr=NULL;

GoToHead();

while (curr!=NULL) {

if (curr->data==key) {

node \*t= new node();

t->data= val;

t->next= NULL;

t->next= curr;

ptr->next= t;

length++;

break;

}

ptr= curr;

curr= curr->next;

}

}

void printLinklist() {

GoToHead();

while(curr!=NULL) {

cout<<curr->data<<"\t";

curr= curr->next;

}

}

void DeleteNodeUsingKey(int key) {

GoToHead();

node \*prenode= new node();

if(curr->data== key) {

head= curr->next;

delete curr;

length--;

return;

} else

while(curr!=NULL) {

if(curr->data==key) {

prenode->next= curr->next;

delete curr;

length--;

break;

}

prenode= curr;

curr=curr->next;

}

}

void DeleteNodeUsingPos(int pos) {

GoToHead();

node \*prenode= new node();

if(pos>length) {

cout<<"This Position dosenot exist"<<endl;

return;

} else if (pos==1 ) { // if we want to delet head node

prenode= curr;

head= curr->next;

delete prenode;

length--;

} else {

for (int x=1; x<pos; x++) {

prenode= curr;

curr= curr->next;

}

prenode->next= curr->next;

delete curr;

length--;

}

}

void InsertNodeUsingKey(int val, int key, bool isBefore) {

if (isBefore)

InsertBeforeSpecificKey( val, key);

else

InsertAfterSpecificKey( val, key);

}

void InsertNodeUsingPos(int val, int pos, bool isBefore) {

GoToHead();

if(pos>length) {

cout<<"This Position dosenot exist"<<endl;

return;

} else if (pos==1 && isBefore ) { // if we want to insert before head

AddNodeBeforeHead(val);

} else {

node \*prenode= new node();

for (int x=1; x<pos; x++) {

prenode= curr;

curr= curr->next;

}

if (isBefore) {

node \*t= new node();

t->data= val;

t->next= NULL;

t->next= curr;

prenode->next= t;

} else {

node \*t= new node();

t->data= val;

t->next= NULL;

t->next= curr->next;

curr->next= t;

}

}

}

int main () {

head->data= 1;

head->next=NULL;

insertNodeAtEnd(2);

insertNodeAtEnd(3);

insertNodeAtEnd(4);

printLinklist();

cout<<endl;

InsertAfterSpecificKey(99, 2);

printLinklist();

cout<<endl;

DeleteNodeUsingKey(99);

printLinklist();

cout<<endl;

InsertBeforeSpecificKey(99, 2);

printLinklist();

cout<<endl;

InsertNodeUsingPos(88,1,true);

printLinklist();

cout<<endl;

DeleteNodeUsingPos(1);

DeleteNodeUsingPos(2);

printLinklist();

cout<<endl;

return 0;

}