#include<bits/stdc++.h>

#include "node.h"

using namespace std;

void menu();

void pushFront(int key);

int\* topFront();

void print();

void popFront();

int\* topBack();

void pushBack(int key);

void popBack();

bool empty();

void erase(int key);

bool find(int key);

Node\* findAddr(int key);

void addBefore(Node\* node,int key);

void addAfter(Node\* node,int key);

Node\* head = NULL;

Node\* tail = NULL;

Node\* node1;

int main(){

cout<<"\n";

int choice;

int key;

int\* p;

while(true){

menu();

cout << "Enter choice: ";

cin >> choice;

switch(choice){

case 1:

cout << "Enter key: ";

cin >> key;

pushFront(key);

cout << "operation completed" << endl;

break;

case 2:

popFront();

cout << "operation completed" << endl;

break;

case 3:

p=topFront();

cout << "top front: " << \*p << endl;

cout << "operation completed" << endl;

break;

case 4:

cout << "Enter key: ";

cin >> key;

pushBack(key);

cout << "operation completed" << endl;

break;

case 5:

popBack();

cout << "operation completed" << endl;

break;

case 6:

cout<<\*topBack()<<endl;

cout << "operation completed" << endl;

break;

case 7:

cout << "Enter key: ";

cin >> key;

//if(find(key)==1){

//cout<<"True"<<endl;

//}

//else if(find(key)==0){

//cout<<"False"<<endl;

// }

cout<<find(key)<<endl;

cout << "operation completed" << endl;

break;

case 8:

cout << "Enter key: ";

cin >> key;

erase(key);

cout << "operation completed" << endl;

break;

case 9:

//if(empty()==1){

// cout<<"True"<<endl;

// }

//else{

// cout<<"False"<<endl;

//}

cout<<empty()<<endl;

cout << "operation completed" << endl;

break;

case 10:

int k;

cout << "Enter key: ";

cin >> key;

cout << "Before which key: ";

cin >> k;

node1=findAddr(k);

if(node1==NULL){

cout<<"node is not present"<<endl;

}

else{

addBefore(node1,key);

}

cout << "operation completed" << endl;

break;

case 11:

int a;

cout << "Enter key: ";

cin >> key;

cout << "After which node: ";

cin >> a;

node1=findAddr(a);

addAfter(node1,key);

cout << "operation completed" << endl;

break;

case 12:

print();

cout << "operation completed" << endl;

break;

case 13:

cout << "thank you!" << endl;

exit(0);

}

}

return 0;

}

void menu(){

cout << "------------------" << endl;

cout << "Singly Linked List" << endl;

cout << "------------------" << endl;

cout << " (1) push front ... (2) pop front ... (3) top front ..." << endl;

cout << " (4) push back .... (5) pop back .... (6) top back ...." << endl;

cout << " (7) find key ..... (8) erase key ... (9) empty ......." << endl;

cout << "(10) add before .. (11) add after .. (12) print ......." << endl;

cout << "(13) exit ..." << endl;

cout << "------------------" << endl;

}

void pushFront(int key){

Node\* node = new Node(key);

node->next = head;

head = node;

if(tail == NULL)

tail = head;

}

int\* topFront(){

return head->key;

}

void print(){

Node\* p = head;

if(head==NULL){

cout << "List is empty" << endl;

return;

}

while(p!= NULL){

cout << \*(p->key) << " ";

p = p->next;

}

cout << endl;

}

void popFront(){

if(head==NULL){

cout<<"List is empty"<<endl;

}

head=head->next;

}

void pushBack(int key){

Node\* node=new Node;

node->next=NULL;

node->key=new int(key);

if(tail->next==NULL){

tail->next=node;

}

else{

tail->next=node;

}

}

int\* topBack(){

return tail->key;

}

void popBack(){

Node\* p=head;

if(head==NULL){

cout<<"List is empty"<<endl;

}

while(p->next->next!=NULL){

p=p->next;

}

p->next=NULL;

//delete p;

}

bool empty(){

if(head==NULL){

return true;

}

else{

return false;

}

}

void erase(int key){

Node\* temp;

if(\*(topFront())==(key)){

temp=head;

head=head->next;

delete temp;

}

else{

Node\* p=head;

Node\* node=new Node;

while(p!=NULL){

if(\*(p->next->key)==key){

temp=p->next;

p->next=p->next->next;

delete temp;

break;

}

p=p->next;

}

}

//Node\* p=head;

//Node\* node=new Node;

//node->key=new int(key);

//while(p!=NULL){

//if(\*(p->key)==\*(node->key)){

// \*(p->key)=NULL;

// delete p;

//}

// p=p->next;

//}

}

bool find(int key){

if(head==NULL){

cout<<"list is empty"<<endl;

//break;

}

bool found=false;

Node\* p=head;

Node\* node=new Node;

node->key=new int(key);

while(p!=NULL){

if(\*(p->key)==\*(node->key)){

found=true;

break;

}

p=p->next;

}

return found;

}

Node\* findAddr(int key){

Node\* p=head;

while(p!=NULL){

if(\*(p->key)==(key)){

return p;

break;

}

p=p->next;

}

}

void addBefore(Node\* node,int key){

Node\* node2=new Node(key);

Node\* p=head;

if(head!=node){

while(p!=NULL){

if(p->next==node){

node2->next=p->next;

p->next=node2;

break;

}

p=p->next;

}

}

else{

pushFront(key);

}

}

void addAfter(Node\* node,int key){

Node\* newnode=new Node;

newnode->key=new int(key);

newnode->next=NULL;

newnode->next=node->next;

node->next=newnode;

}