#include <iostream>

using namespace std;

#define BEGIN 1

#define END 0

struct Spis2 {

int info;

Spis2\* prev, \* next;

} \*begin\_, \*end\_;

//------------------- Создание первого элемента -----------------------------

void Create\_Spis2(Spis2\*\* b, Spis2\*\* e, int in) {

Spis2 \*t = new Spis2;

t->info = in;

t->next = t->prev = NULL;

\*b = \*e = t;

}

//------------------- Добавление элемента в список --------------------------

void Add\_Spis2\_In\_Begin(Spis2\*\* b, Spis2\*\* e, int in) {

Spis2\* t = new Spis2;

t->info = in;

t->prev = NULL;

t->next = \*b;

(\*b)->prev = t;

\*b = t;

}

void Add\_Spis2\_In\_End(Spis2\*\* b, Spis2\*\* e, int in) {

Spis2\* t = new Spis2;

t->info = in;

t->next = NULL;

t->prev = \*e;

(\*e)->next = t;

\*e = t;

}

//--------------------- Просмотр элементов списка ---------------------------

void View\_Spis2(int kod, Spis2\* t) {

while (t != NULL) {

cout << t->info << endl;

if (kod == END) t = t->next;

else t = t->prev;

}

}

void Del\_All(Spis2\*\* p) {

Spis2\* t;

while (\*p != NULL) {

t = \*p;

\*p = (\*p)->next;

delete t;

}

}

void Sort\_p(Spis2 \*\* p)

{

Spis2\* t = NULL, \* t1, \* r;

if ((\*p)->next->next == NULL) return;

do {

for (t1 = \*p; t1->next->next != t; t1 = t1->next)

if (t1->next->info > t1->next->next->info) {

r = t1->next->next;

t1->next->next = r->next;

r->next = t1->next;

t1->next = r;

}

t = t1->next;

} while ((\*p)->next->next != t);

}

void Sort\_info(Spis2\* p)

{

Spis2\* t = NULL, \* t1;

int r;

do {

for (t1 = p; t1->next != t; t1 = t1->next)

if (t1->info > t1->next->info) {

r = t1->info;

t1->info = t1->next->info;

t1->next->info = r;

}

t = t1;

} while (p->next != t);

}

Spis2\* delete\_elements(Spis2\* p)

{

bool is\_even = false;

Spis2\* t = NULL;

while (p != NULL)

{

if (is\_even)

{

Spis2\* for\_delete = p;

p = p->next;

delete for\_delete;

}

else

{

Spis2\* s\_2 = p;

p = p->next;

s\_2->next = t;

t = s\_2;

}

is\_even = !is\_even;

}

return t;

}

int main()

{

int size = 0;

cout << "list size: ";

cin >> size;

for (int i = 0; i < size; i++)

{

cout << "element #" << i <<":"<< endl;

if (i == 0) // if is the first element

{

cout << "info: ";

int in\_;

cin >> in\_;

Create\_Spis2(&begin\_, &end\_, in\_);

continue;

}

int flag = 0;

cout << "add to begin or end? 1/0: ";

cin >> flag;

int in;

cout << "info: ";

cin >> in;

cout << endl;

if (flag == BEGIN)

Add\_Spis2\_In\_Begin(&begin\_, &end\_, in);

else

Add\_Spis2\_In\_End(&begin\_, &end\_, in);

}

View\_Spis2(0, begin\_);

Sort\_p(&begin\_);

Sort\_info(begin\_);

cout << "after sorting:" << endl;

cout << endl;

View\_Spis2(END, begin\_);

begin\_ = delete\_elements(begin\_);

cout << "after deleting:" << endl;

View\_Spis2(END, begin\_);

Del\_All(&begin\_);

return 0;

}

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

#include <iostream>

#include <iomanip>

using namespace std;

struct Node {

int data;

Node\* prev;

};

struct List {

Node\* begin;

Node\* top;

int amount;

};

List\* createList();

void pushToTheBegin(List\*, int);

void pushToTheEnd(List\*, int);

int pop(List\*);

void sort(List\*);

void printListFromTheBegin(List\*);

void printListFromTheEnd(List\*);

void deleteList(List\*\*);

List\* createList() {

List\* l = new List;

l->begin = nullptr;

l->top = nullptr;

l->amount = 0;

return l;

}

void pushToTheBegin(List\* l, int newNumb) {

Node\* n = new Node;

if (l->top == nullptr) {

n->prev = l->top;

n->data = newNumb;

l->top = n;

l->begin = n;

l->amount++;

return;

}

if (l->top->prev == nullptr) {

n->prev = l->top->prev;

l->top->prev = n;

l->begin = n;

n->data = newNumb;

l->amount++;

return;

}

Node\* temp = l->top;

while (temp->prev != l->begin) {

temp = temp->prev;

}

n->prev = temp->prev->prev;

temp->prev->prev = n;

l->begin = n;

n->data = newNumb;

l->amount++;

}

void pushToTheEnd(List\* l, int newNumb) {

Node\* n = new Node;

n->prev = l->top;

n->data = newNumb;

l->top = n;

l->amount++;

if (l->amount == 1) {

l->begin = n;

}

}

int pop(List\* l) {

if (l->top == nullptr) {

cout << "В списке отсутствуют элементы";

return 1;

}

else {

Node\* n = l->top;

while (n->prev != l->begin) {

n = n->prev;

}

l->begin = n;

int numb = n->prev->data;

delete n->prev;

l->amount--;

return numb;

}

}

void sort(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return;

}

Node\* curr = l->top;

Node\* temp = nullptr;

int tempAmount = l->amount;

if (tempAmount % 2 == 0) {

l->top = curr->prev;

delete curr;

curr = l->top;

tempAmount--;

l->amount--;

}

while (tempAmount > 1) {

temp = curr->prev;

curr->prev = curr->prev->prev;

delete temp;

curr = curr->prev;

l->amount--;

tempAmount -= 2;

}

}

void printListFromTheBegin(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return;

}

Node\* n = l->top;

int count = l->amount - 1;

for (int i = count; i > 0; i--) {

int temp = 1;

while (temp <= i) {

n = n->prev;

temp++;

}

cout << n->data << setw(15);

n = l->top;

}

cout << n->data << setw(15) << endl;

}

void printListFromTheEnd(List\* l) {

if (l == nullptr)

{

cout << endl << "Стек пустой";

return;

}

Node\* n = l->top;

while (n != nullptr) {

cout << n->data << setw(15);

n = n->prev;

}

cout << endl;

}

void deleteList(List\*\* l) {

Node\* last = (\*l)->top;

Node\* second = nullptr;

while (last != nullptr) {

if (last->prev == nullptr) {

(\*l)->begin = nullptr;

}

second = last;

last = last->prev;

delete second;

(\*l)->amount--;

}

delete \*l;

\*l = nullptr;

}

int main() {

setlocale(LC\_ALL, "Russian");

List\* l = createList();

int temp = 0;

while (temp != 1) {

int k, numb;

cout << "Добавить число в: 1 - начало очереди, 2 - конец очереди: ";

cin >> k;

switch (k) {

case 1:

cout << "Введите число: ";

cin >> numb;

pushToTheBegin(l, numb);

break;

case 2:

cout << "Введите число: ";

cin >> numb;

pushToTheEnd(l, numb);

break;

default:

cout << "Неверный ввод";

return 1;

}

cout << "Введите любое число, если хотите продолжить, или 1, если хотите закончить: ";

cin >> temp;

}

int c;

cout << "Вы хотите увидеть очередь: 1 - с начала, 2 - с конца: ";

cin >> c;

switch (c) {

case 1:

printListFromTheBegin(l);

break;

case 2:

printListFromTheEnd(l);

break;

default:

cout << "Неверный ввод";

return 1;

}

sort(l);

int m;

cout << "Вы хотите увидеть отсортированную(все четные элементы удалены) очередь: 1 - с начала, 2 - с конца: ";

cin >> m;

switch (m) {

case 1:

printListFromTheBegin(l);

break;

case 2:

printListFromTheEnd(l);

break;

default:

cout << "Неверный ввод";

return 1;

}

deleteList(&l);

return 0;

}