

#include <iostream>

#include <fstream>

#include <conio.h>

#include <math.h>

using namespace std;

ifstream input("input.txt");

ofstream output("output.txt");

struct Node

{

int data;

struct Node\* Next;

};

typedef struct Node NODE;

struct List

{

NODE\* Head = NULL, \* Tail = NULL;

};

typedef struct List LIST;

NODE\* GetNode(int DATA) {

NODE\* p = new NODE;

if (p==NULL)

{

cout << "\n Khong du dung luong cap phat con tro!";

return NULL;

}

p->data = DATA;

p->Next = NULL;

return p;

}

void AddHead(LIST& l, NODE\* p) {

if (l.Head == NULL)

{

l.Head = l.Tail = p;

}

else {

p->Next = l.Head;

l.Head = p;

}

}

void AddTail(LIST& l,NODE \*p) {

if (l.Head==NULL)

{

l.Head = l.Tail = p;

}

else {

l.Tail->Next = p;

l.Tail = p;

}

}

void Input(LIST& l) {

int a[100];

int n;

input >> n;

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

{

input >> a[i];

NODE\* p;

p = GetNode(a[i]);

AddTail(l, p);

}

cout << "Nhap File Thanh Cong";

}

void OutPut(LIST l) {

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

output << p->data << " ";

}

output << "\n";

cout << "\n";

}

int TinhTong(LIST l) {

int tong = 0;

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

tong += p->data;

}

cout << "\nTinh Tong Thanh Cong!";

output << "\n";

cout << "\n";

return tong;

}

void LietKeCacSoChan(LIST l) {

int count=0;

output << "\n\nDanh sach cac NODE so chan: \n";

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

if (p->data%2==0)

{

output << p->data << " ";

count++;

}

}

if (count==0)

{

cout << "\nDanh sach node khong co so chan";

}

else {

cout << "\nLiet ke cac so chan thanh cong!";

}

cout << "\n";

}

void LietKeCacSoLe(LIST l) {

int count = 0;

output << "\n\nDanh sach cac NODE so le: \n";

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

if (p->data % 2 != 0)

{

output << p->data << " ";

count++;

}

}

if (count == 0)

{

cout << "\nDanh sach node khong co so le";

}

else {

cout << "\nLiet ke cac so le thanh cong!";

}

output << "\n";

cout << "\n";

}

void HoanVi(int &data1, int &data2) {

int tem = data1;

data1 = data2;

data2 = tem;

}

void SapXepDSNodeTangDan(LIST &l){

for (NODE\* p = l.Head; p != l.Tail;p=p->Next) {

for (NODE\* q = p->Next; q != NULL; q = q->Next) {

if (p->data > q->data) {

HoanVi(p->data, q->data);

}

}

}

cout << "\nSap xep tang dan thanh cong!";

output << "\n";

cout << "\n";

}

void SapXepDSNodeGiamDan(LIST& l) {

for (NODE\* p = l.Head; p != l.Tail; p = p->Next) {

for (NODE\* q = p->Next; q != NULL; q = q->Next) {

if (p->data < q->data) {

HoanVi(p->data, q->data);

}

}

}

cout << "Sap xep giam dan thanh cong!";

}

bool IsSoNguyenTo(int x) {

int count = 0;

for (int i = 2; i<x; i++) {

if (x % i == 0) {

return false;

}

}

return true;

}

void LietKeCacSoNguyenTo(LIST l) {

int count = 0;

output << "\nDanh sach cac NODE so nguyen to(Chi chia het cho 1 va chinh no): \n";

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

if (IsSoNguyenTo(p->data) ){

output << p->data << " ";

count++;

}

}

if (count == 0)

{

cout << "\nDanh sach node khong co so nguyen to";

}

else {

cout << "\nLiet ke cac so nguyen to thanh cong!";

}

output << "\n";

cout << "\n";

}

bool IsSoChinhPhuong(int x) {

int can = sqrt(x);

if (can \* can == x) {

return true;

}

return false;

}

void LietKeCacSoChinhPhuong(LIST l) {

int count = 0;

output << "\nDanh sach cac NODE so chinh phuong(Co can bac 2 la 1 so nguyen): \n";

for (NODE\* p = l.Head; p != NULL; p = p->Next) {

if (IsSoChinhPhuong(p->data)) {

output << p->data << " ";

count++;

}

}

if (count == 0)

{

cout << "\nDanh sach node khong co so chinh phuong";

}

else {

cout << "\nLiet ke cac so chinh phuong thanh cong!";

}

output << "\n";

cout << "\n";

}

int main() {

LIST l;

Input(l);

output << "Danh sach cac NODE: \n";

OutPut(l);

output << "Tong cua cac node la: "<<TinhTong(l);

LietKeCacSoChan(l);

LietKeCacSoLe(l);

output << "\nDanh sach cac NODE sau khi sap xep tang dan:";

SapXepDSNodeTangDan(l);

OutPut(l);

output << "\nDanh sach cac NODE sau khi sap xep giam dan:"<<endl;

SapXepDSNodeGiamDan(l);

OutPut(l);

LietKeCacSoNguyenTo(l);

LietKeCacSoChinhPhuong(l);

return 0;

}