#include <iostream>

#include <vector>

using namespace std;

struct CPoint {

CPoint(int x = 0, int y = 0) : x(x), y(y) {}

bool operator<(const CPoint &rhs) const {

if (x != rhs.x) {

return x < rhs.x;

} else return y < rhs.y;

}

int x;

int y;

};

template<class Iterator>

void InsertionSort(Iterator begin, Iterator end) {

for (Iterator i = begin + 1; i != end; ++i) {

CPoint value = \*(i);

Iterator sortedPtr = i - 1;

for (; sortedPtr != begin && value < \*(sortedPtr); --sortedPtr) {

\*(sortedPtr + 1) = \*sortedPtr;

}

\*(sortedPtr + 1) = value;

}

}

int main() {

int n = 0;

vector<CPoint> a;

cin >> n;

CPoint point;

int i = 1;

a.push\_back(0);

while (i < n + 1) {

cin >> point.x >> point.y;

a.push\_back(point);

i++;

}

InsertionSort(a.begin(), a.end());

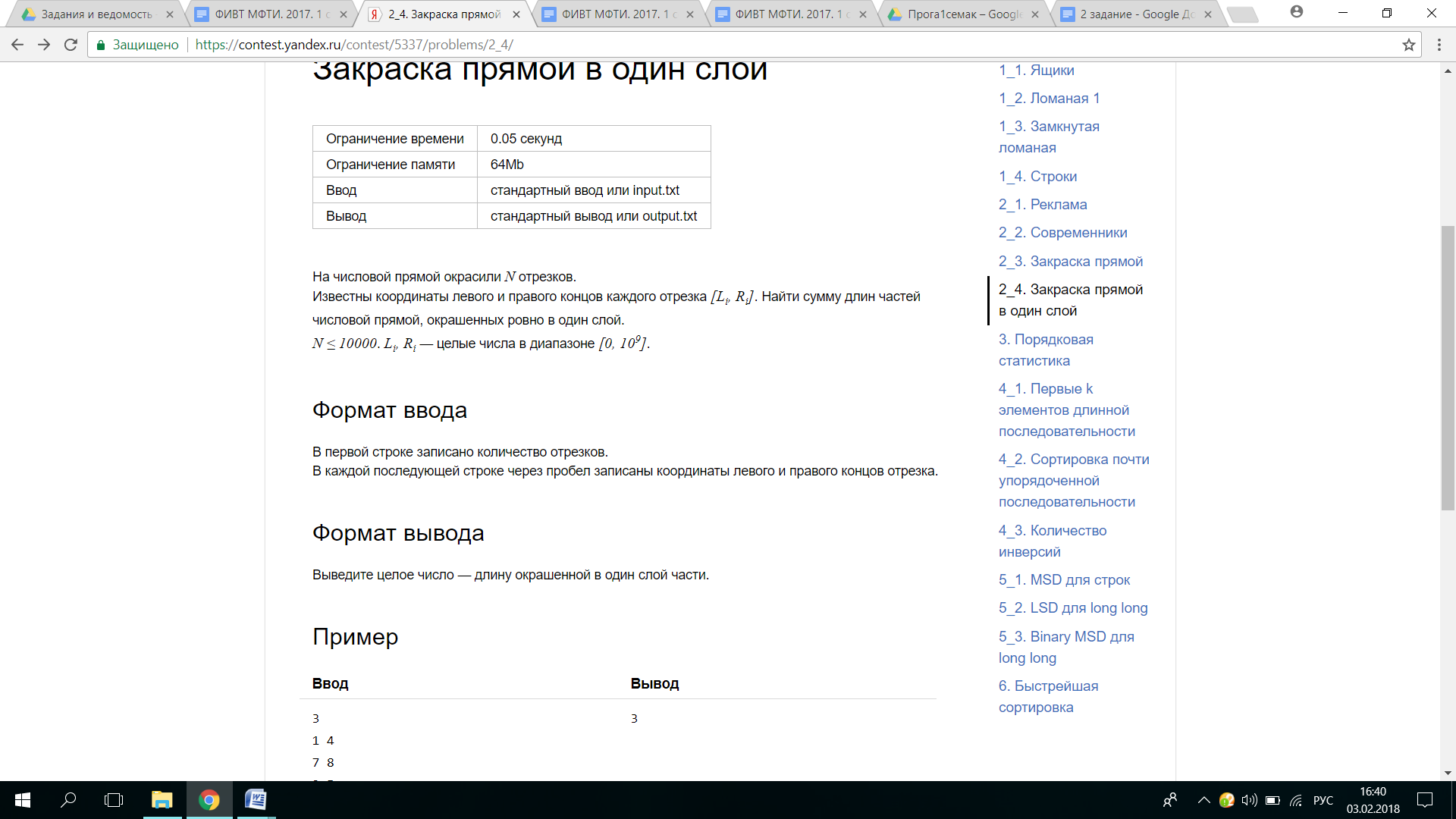
for (int i = 1; i < n + 1; i++) {

cout << a[i].x << " " << a[i].y << endl;

}

return 0;

}



#include <iostream>

#include <vector>

using namespace std;

struct CPoint {

CPoint(int value = 0, int index = 0) : value(value), index(index) {}

bool operator<(const CPoint &rhs) const {

if (value != rhs.value) {

return value < rhs.value;

} else return index < rhs.index;

}

int value;

int index;

};

template<class Iterator>

void SiftDown(Iterator begin, Iterator i, Iterator end) {

Iterator left = begin + 2 \* (i - begin) + 1;

Iterator right = begin + 2 \* (i - begin) + 2;

Iterator largest = i;

if (left < end && \*(i) < \*(left))

largest = left;

if (right < end && \*(largest) < \*(right))

largest = right;

if (largest != i) {

std::swap(\*(i), \*(largest));

SiftDown(begin, largest, end);

}

}

template<class Iterator>

void BuildHeap(Iterator begin, Iterator i, Iterator end) {

for (Iterator i = begin + (end - begin) / 2 - 1; i >= begin; --i) {

SiftDown(begin, i, end);

}

}

template<class Iterator>

void HeapSort(Iterator begin, Iterator end) {

Iterator heapSize = end;

BuildHeap(begin, heapSize, end);

while (heapSize > begin + 1) {

swap(\*(begin), \*(heapSize - 1));

--heapSize;

SiftDown(begin, begin, heapSize);

}

}

template<class Iterator>

int OneLayer(Iterator begin, Iterator end) {

int s = 0;

int start = 0, last = 0, lenth = 0;

for (Iterator i = begin; i < end; i++) {

CPoint point = \*(i);

if (point.index == 0) {

s++;

} else if (point.index == 1) {

s--;

}

if (s == 1) {

start = point.value;

} else if (start != 0) {

last = point.value;

lenth = lenth + last - start;

start = 0;

}

}

return lenth;

}

int main() {

int n = 0;

vector<CPoint> a;

cin >> n;

n = 2 \* n;

CPoint point;

for (int i = 0; i < n; i = i + 2) {

cin >> point.value;

point.index = 0;

a.push\_back(point);

cin >> point.value;

point.index = 1;

a.push\_back(point);

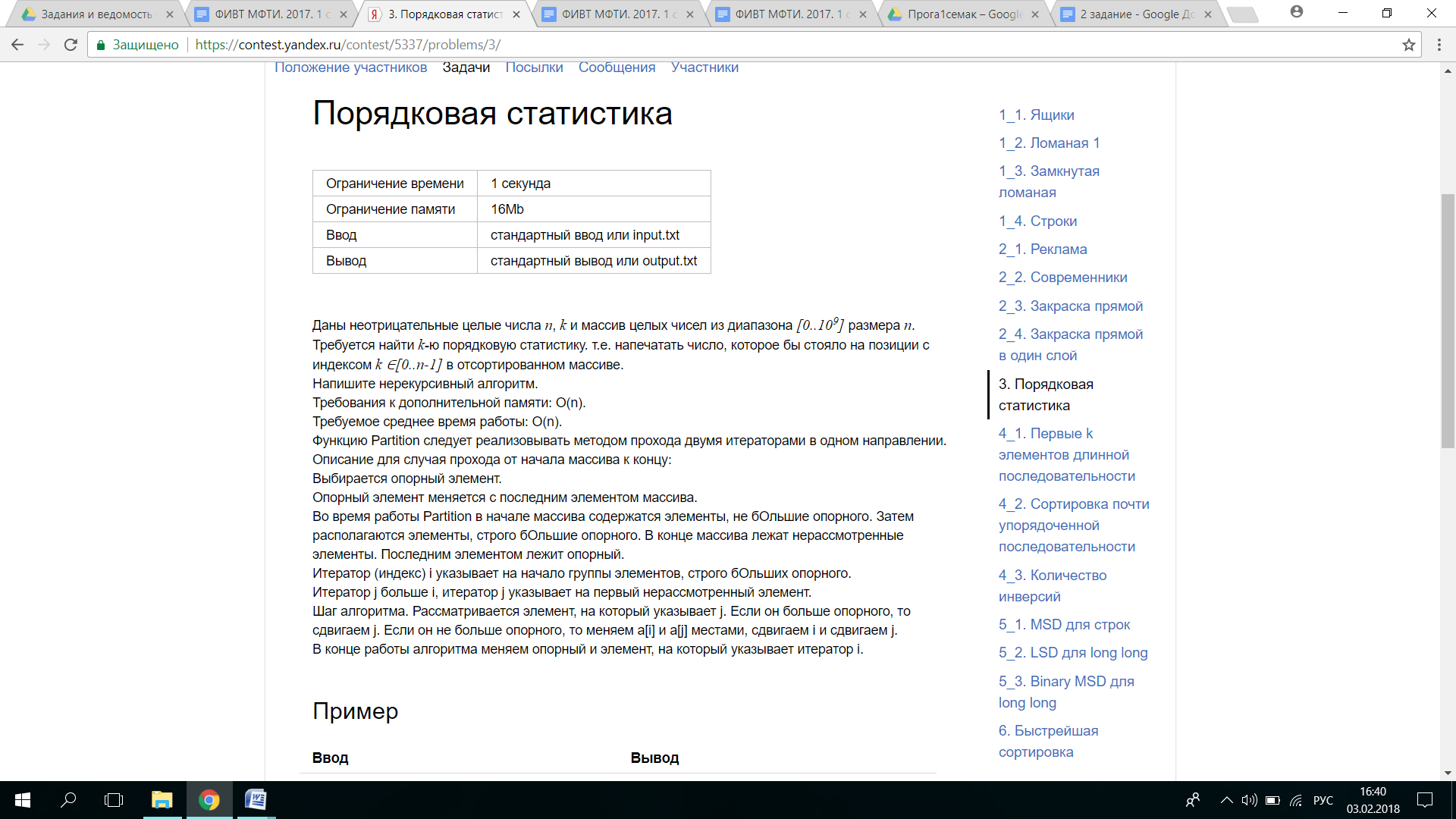
}

HeapSort(a.begin(), a.end());

cout << OneLayer(a.begin(), a.end());;

return 0;

}

#include <iostream>

#include <vector>

#include <stack>

#include <cstdlib>

using namespace std;

template<class Iterator>

void MyMedianaOfThree(Iterator a, Iterator b, Iterator c, Iterator end) {

int aValue = \*(a);

int bValue = \*(b);

int cValue = \*(c);

if (aValue >= bValue) {

if (cValue >= aValue) {

swap(\*a, \*(end - 1));

} else {

if (bValue >= cValue) swap(\*b, \*(end - 1));

else swap(\*c, \*(end - 1));

}

} else {

if (cValue >= bValue) { swap(\*b, \*(end - 1)); }

else {

if (aValue >= cValue) swap(\*a, \*(end - 1));

else swap(\*c, \*(end - 1));

}

}

}

template<class Iterator>

void MyMediana(Iterator begin, Iterator end) {

Iterator One = begin + (end - begin) / 3;

Iterator Two = begin + (end - begin) / 2;

Iterator Three = begin + (end - begin) \* 2 / 3;

MyMedianaOfThree(One, Two, Three, end);

}

template<class Iterator>

Iterator Partition(Iterator begin, Iterator end) {

if (end <= begin + 1) {

return begin;

}

const int &pivot = \*(end - 1);

Iterator i = begin;

Iterator j = i;

while (j < end - 1) {

if (\*(j) > pivot) {

j++;

} else {

swap(\*(i), \*(j));

i++;

j++;

}

}

swap(\*(i), \*(end - 1));

return i;

}

template<class Iterator>

int QuickSort(Iterator begin, Iterator l, Iterator r, int k) {

stack<Iterator> s;

s.push(r);

s.push(l);

while (!s.empty()) {

l = s.top();

s.pop();

r = s.top();

s.pop();

if (r <= l) continue;

MyMediana(l, r);

Iterator i = Partition(l, r);

if (k == i - begin) {

return \*(i);

} else if (k > i - begin) {

s.push(r);

s.push(i + 1);

} else {

s.push(i);

s.push(l);

}

}

}

int main() {

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

int n = 0, k = 0;

vector<int> a;

cin >> n;

cin >> k;

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

int m;

cin >> m;

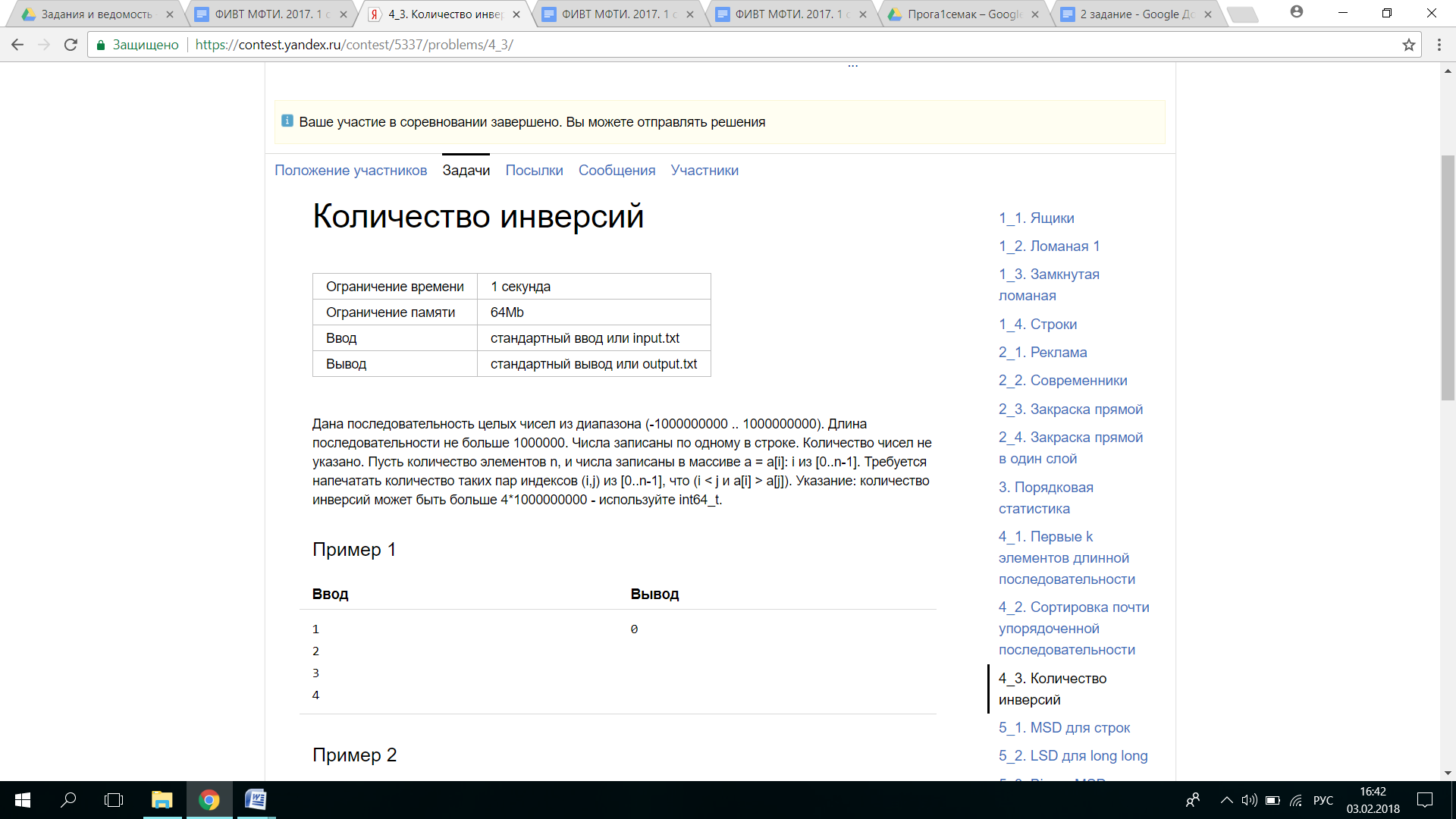
a.push\_back(m);

}

cout << QuickSort(a.begin(), a.begin(), a.end(), k);

return 0;

}

#include <iostream>

#include <cstring>

#include <vector>

#include <stdio.h>

using namespace std;

template<class Iterator>

void Merge(Iterator a, int aLen, Iterator b, int bLen, Iterator c, int64\_t &inv) {

int i = 0, j = 0;

while (i < aLen && j < bLen) {

if (\*(a + i) <= \*(b + j)) {

\*(c + i + j) = \*(a + i);

++i;

} else {

inv = inv + aLen - i;

\*(c + i + j) = \*(b + j);

++j;

}

}

if (i == aLen) {

for (; j < bLen; ++j)

\*(c + i + j) = \*(b + j);

} else {

for (; i < aLen; ++i)

\*(c + i + j) = \*(a + i);

}

}

template<class Iterator>

void MergeSort(Iterator a, int N, int64\_t &inv) {

if (N <= 1) {

return;

}

int firstLen = N / 2;

int secondLen = N - firstLen;

MergeSort(a, firstLen, inv);

MergeSort(a + firstLen, secondLen, inv);

vector<int64\_t> c(N);

c.push\_back(0);

Merge(a, firstLen, a + firstLen, secondLen, c.begin(), inv);

for (int k = 0; k < N; k++) {

\*(a + k) = c[k];

}

}

int main() {

int i = 0;

int64\_t n;

int64\_t inv = 0;

vector<int64\_t> a;

while (scanf("%d", &n) != EOF) {

i++;

a.push\_back(n);

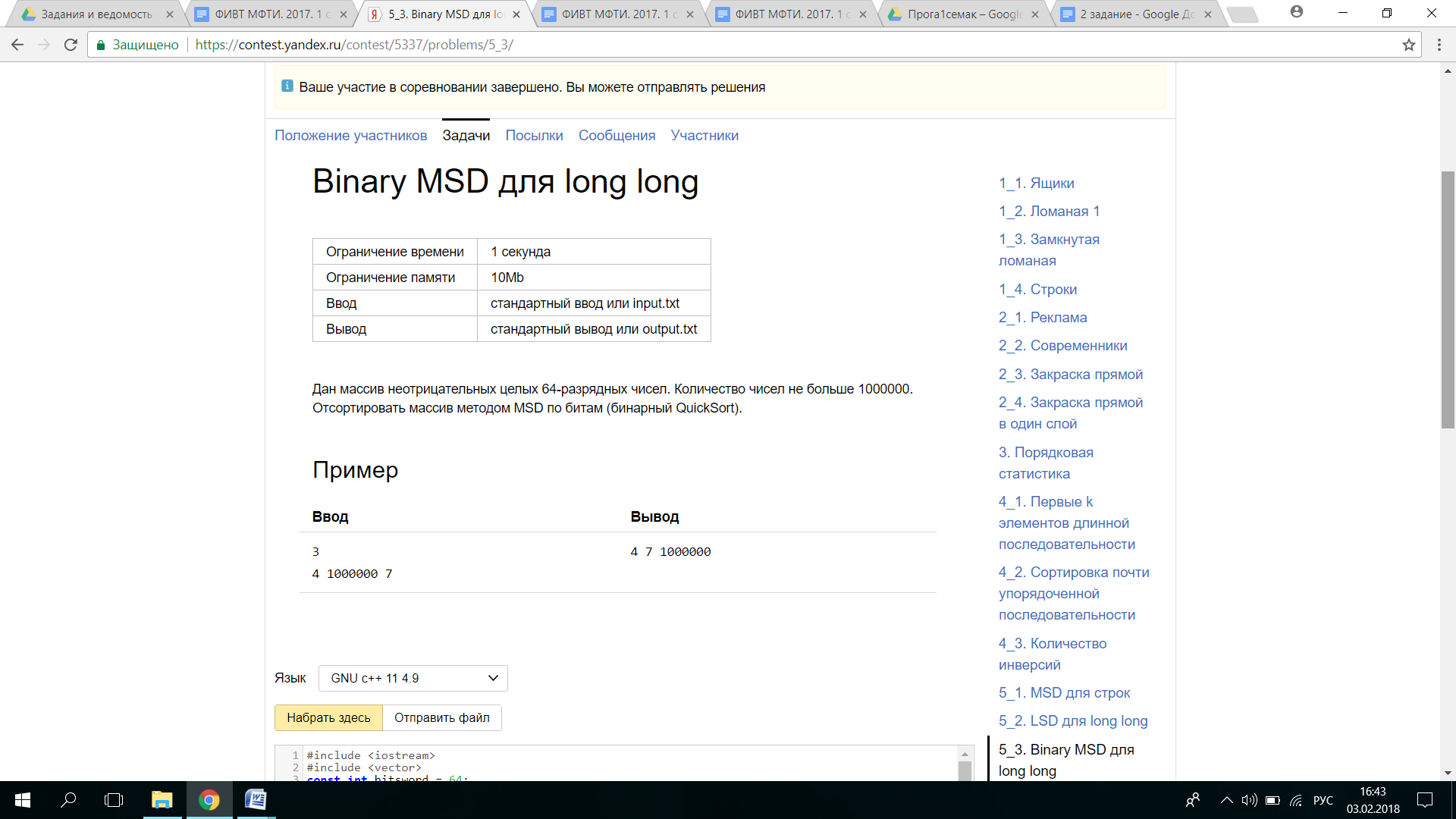
}

MergeSort(a.begin(), i, inv);

cout << inv;

return 0;

}



#include <iostream>

#include <vector>

const int bitsword = 64;

using namespace std;

inline int64\_t digit(long long A, int B) {

return ((A >> (bitsword - B - 1)) & 1);

}

template <class Iterator>

void quicksort( Iterator a,Iterator l, Iterator r, int d) {

Iterator i = l, j = r;

if ((r-a <= l-a) || (d > bitsword)) return;

while (j-a != i-a) {

while (digit(\*(i), d) == 0 && (i-a < j-a)) i++;

while (digit(\*(j), d) == 1 && (j-a > i-a)) j--;

swap(\*(i), \*(j));

}

if (digit(\*(r), d) == 0) j++;

quicksort(a, l, j - 1, d + 1);

quicksort(a, j, r, d + 1);

}

int main() {

int n;

long long m;

cin >> n;

vector <long long> a;

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

cin >> m;

a.push\_back(m);

}

quicksort(a.begin(),a.begin(), a.end()-1, 0);

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

cout << a[i] << " ";

}

return 0;

}

БЫСТРЕЙШАЯ СОРТИРОВКА

#include <iostream>

#include <vector>

#include <stack>

#include <cstdlib>

#include <ctime>

#include <cstring>

#include <stdio.h>

using namespace std;

template<class Iterator>

void MyMedianaOfThree(Iterator a, Iterator b, Iterator c, Iterator end) {

int aValue = \*(a);

int bValue = \*(b);

int cValue = \*(c);

if (aValue >= bValue) {

if (cValue >= aValue) {

swap(\*a, \*(end - 1));

} else {

if (bValue >= cValue) swap(\*b, \*(end - 1));

else swap(\*c, \*(end - 1));

}

} else {

if (cValue >= bValue) { swap(\*b, \*(end - 1)); }

else {

if (aValue >= cValue) swap(\*a, \*(end - 1));

else swap(\*c, \*(end - 1));

}

}

}

template<class Iterator>

void MyMediana(Iterator begin, Iterator end) {

Iterator One = begin + (end - begin) / 3;

Iterator Two = begin + (end - begin) / 2;

Iterator Three = begin + (end - begin) \* 2 / 3;

MyMedianaOfThree(One, Two, Three, end);

}

template<class Iterator>

Iterator Partition(Iterator begin, Iterator end) {

if (end <= begin + 1) {

return begin;

}

const int &pivot = \*(end - 1);

Iterator i = begin;

Iterator j = i;

while (j < end - 1) {

if (\*(j) > pivot) {

j++;

} else {

swap(\*(i), \*(j));

i++;

j++;

}

}

swap(\*(i), \*(end - 1));

return i;

}

template<class Iterator>

void QuickSort(Iterator begin, Iterator l, Iterator r) {

stack<Iterator> s;

s.push(r);

s.push(l);

while (!s.empty()) {

l = s.top();

s.pop();

r = s.top();

s.pop();

if (r <= l) continue;

MyMediana(l, r);

Iterator i = Partition(l, r);

if (i - begin - 1 > r - i) {

s.push(i);

s.push(l);

s.push(r);

s.push(i + 1);

} else {

s.push(r);

s.push(i + 1);

s.push(i);

s.push(l);

}

}

}

int main() {

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

srand(time(0));

int n = 0, k = 0, i = 0;

vector<int> a;

while (scanf("%d", &n) != EOF) {

k++;

a.push\_back(n);

}

MyMediana(a.begin(), a.end());

QuickSort(a.begin(), a.begin(), a.end());

for (int i = 9; i < k; i = i + 10) {

cout << a[i] << " ";

}

return 0;

}