# MaratonaCIn

## SELETIVA 2020

# STANDARD TEMPLATE LIBRARY
# ( STL )

## Aula 1

# STANDARD TEMPLATE LIBRARY ( STL )

- Estruturas de dados e algoritmos já implementados.
- Escrever códigos curtos e mais rápidos.
- Evitar bugs desnecessários.

# STANDARD TEMPLATE LIBRARY ( STL )

## Bibliotecas

- vector
- string
- set
- map

- stack
- queue
- algorithm

...

# STANDARD TEMPLATE LIBRARY ( STL )

Bibliotecas

<bits/stdc++.h>

# STANDARD TEMPLATE LIBRARY ( STL )

## Vector

```
vector<tipo> nome;

vector<tipo> nome(tamanho);

vector<tipo> nome(tamanho, valor inicial);
```

# STANDARD TEMPLATE LIBRARY ( STL )

## Vector

- v.push_back(X); // Insere o elemento X no fim do vector
- v.resize(N); // Altera o tamanho do vector para N
- v.clear(); // Reinicia o vector
- v.size(); // Retorna quantia de elementos

# STANDARD TEMPLATE LIBRARY ( STL )

## Vector

```
for(int i=0; i<V.size(); i++) { V[i] … }
```

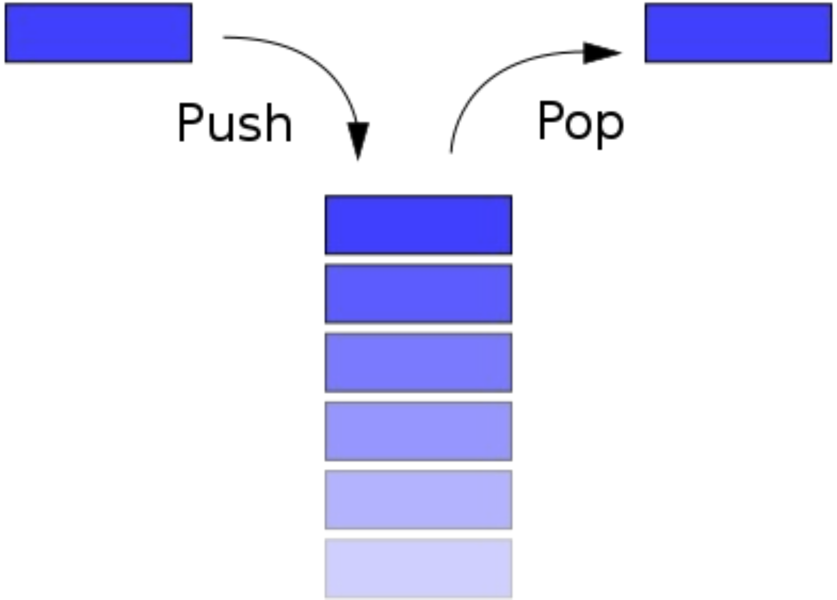
=

```
for(auto u: V) { u … }
```

# STANDARD TEMPLATE LIBRARY ( STL )

## Stack

```
stack<tipo> nome;
```



# STANDARD TEMPLATE LIBRARY ( STL )

## Stack

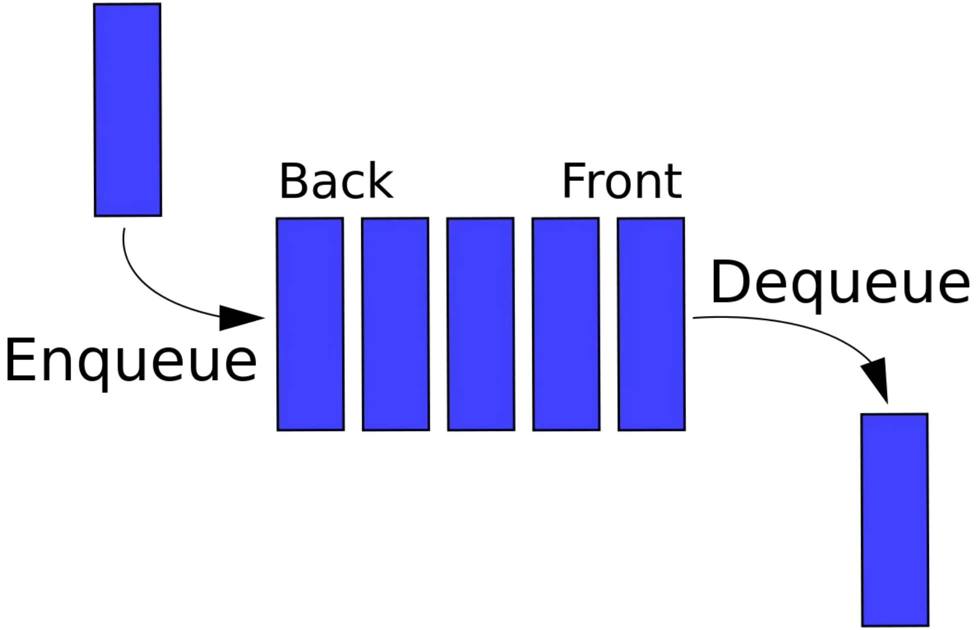
- s.push(X); // Insere o elemento X no topo da pilha
- s.top(); // Retorna o elemento do topo da pilha
- s.pop(); // Retira o elemento do topo da pilha
- s.empty(); // Retorna se a pilha está vazia

# STANDARD TEMPLATE LIBRARY ( STL )

MaratonaCIn

## Queue

```
queue<tipo> nome;
```



# STANDARD TEMPLATE LIBRARY ( STL )

MaratonaCIn

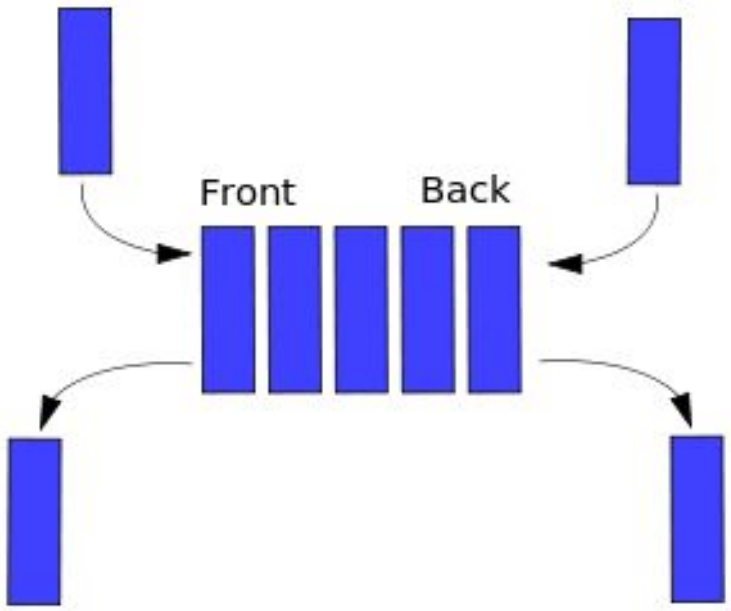
## Queue

- q.push(X); // Insere o elemento X no final da fila
- q.front(); // Retorna o elemento da frente da fila
- q.pop(); // Retira o elemento da frente da fila
- q.empty(); // Retorna se a fila está vazia

# STANDARD TEMPLATE LIBRARY ( STL )

## Deque

```
deque<tipo> nome;
```



# STANDARD TEMPLATE LIBRARY ( STL )

## Deque

- dq.push_front(X);    // Insere elemento na frente
- dq.push_back(X);    // Insere elemento atrás
- dq.front();    // Retorna o elemento da frente
- dq.back();    // Retorna o elemento atrás

# STANDARD TEMPLATE LIBRARY ( STL )

## Deque

- dq.pop_front(); // Retira o elemento da frente
- dq.pop_back(); // Retira o elemento de trás
- dq.size(); // Retorna a quantia de elementos na fila duplamente terminada

# STANDARD TEMPLATE LIBRARY ( STL )

## Priority Queue

```
priority_queue<tipo> nome;

priority_queue<tipo, vector<tipo>, greater<tipo>> nome;

priority_queue<tipo, vector<tipo>, decltype(&funcao)>

nome(funcao);
```

# STANDARD TEMPLATE LIBRARY ( STL )

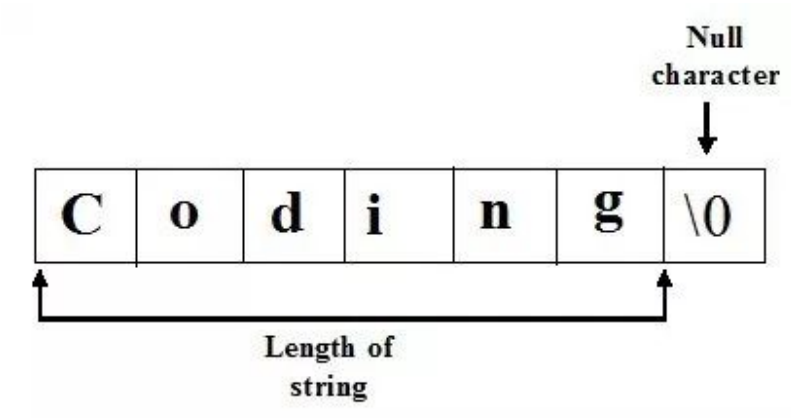
## Priority Queue

- pq.push(X);    // Insere o elemento X na fila prioritária
- pq.top();    // Retorna o maior elemento da fila
- pq.pop();    // Retira o maior elemento da fila
- pq.empty();    // Retorna se a fila está vazia

# STANDARD TEMPLATE LIBRARY ( STL )

## String

```
string nome = "Coding";

cin >> nome;

cout << nome;

getline(cin, nome);
```



| C | o | d | i | n | g | \0 |
|---|---|---|---|---|---|---|

Null character

Length of string

# STANDARD TEMPLATE LIBRARY ( STL )

## String

- str.size(); // Retorna o tamanho da string
- str.clear(); // Reinicia a string
- str.substr(pos, tam); // Retorna substring
- Várias outras funções

# STANDARD TEMPLATE LIBRARY ( STL )

## String

- str1 + str2 // Concatena as strings
- str1 < str2 // Compara alfabeticamente
- str1 == str2 // Retorna se são iguais

# STANDARD TEMPLATE LIBRARY ( STL )

## Stringstream

```
stringstream nome_ss(stringCarregada);

string str;

while(nome_ss >> str) { … }
```

# STANDARD TEMPLATE LIBRARY ( STL )

MaratonaCIn

## Pair

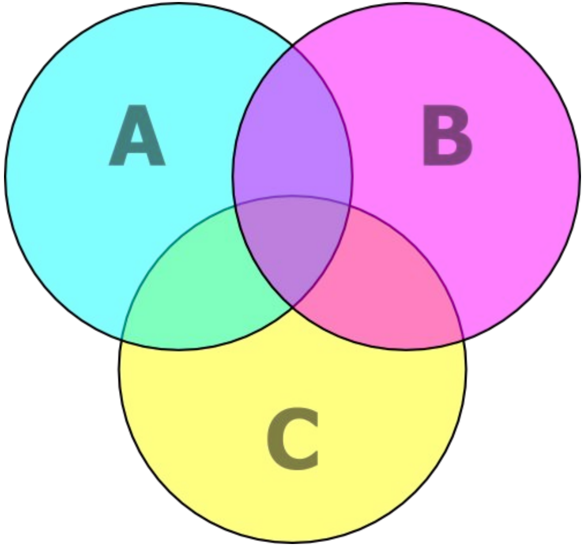
```
pair<tipo1, tipo2> nome;
```

- nome.first // Se refere ao primeiro elemento
- nome.second // Se refere ao segundo elemento

# STANDARD TEMPLATE LIBRARY ( STL )

Set

```
set<tipo> nome;
```



# STANDARD TEMPLATE LIBRARY ( STL )

## Set

- s.insert(X); // Insere o elemento X no conjunto
- s.count(X); // Verifica se o elemento X está presente
- s.clear(); // Reinicia o conjunto
- s.erase(X); // Apaga o valor X do conjunto (funciona)

# STANDARD TEMPLATE LIBRARY ( STL )

## Set

- multiset
- unordered_set

# STANDARD TEMPLATE LIBRARY ( STL )

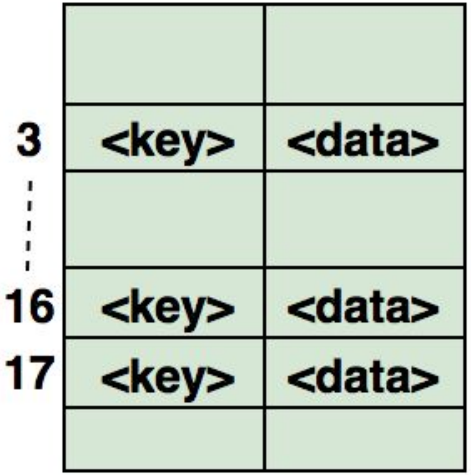
## Set

```
for(auto u: s) {
    u ...    // valor
}
```

# STANDARD TEMPLATE LIBRARY ( STL )

## Map

map<tipo1, tipo2> nome;



| | | |
|---|---|---|
| | | |
| 3 | <key> | <data> |
| | | |
| 16 | <key> | <data> |
| 17 | <key> | <data> |
| | | |

# STANDARD TEMPLATE LIBRARY ( STL )

## Map

- m[chave] = valor   // Atribui o valor à chave
- m.erase(chave);   // Retira a chave e o valor associado
- m.count(chave);   // Verifica se existe a chave X
- m.clear();   // Reinicia o map

# STANDARD TEMPLATE LIBRARY ( STL )

## Map

- multimap
- unordered_map

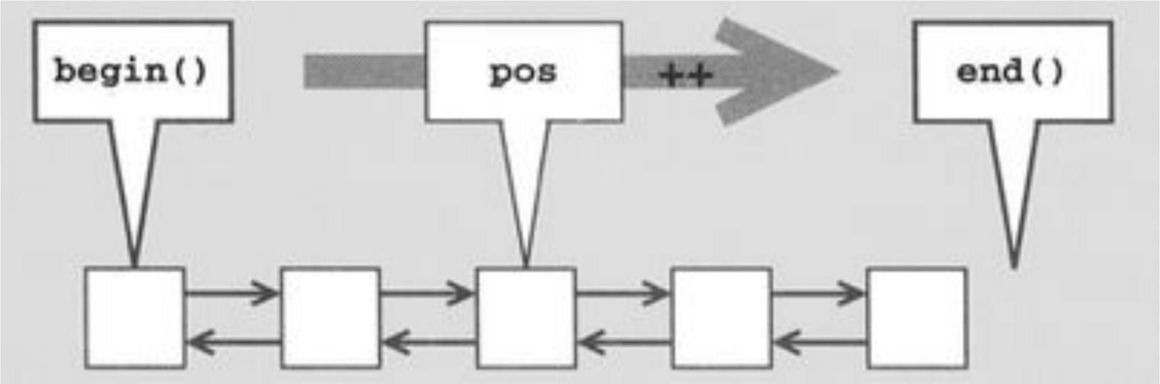
# STANDARD TEMPLATE LIBRARY ( STL )

## Map

```
for(auto u: m) {
    u.first …       // chave
    u.second …   // valor
}
```

# STANDARD TEMPLATE LIBRARY ( STL )

Iterator



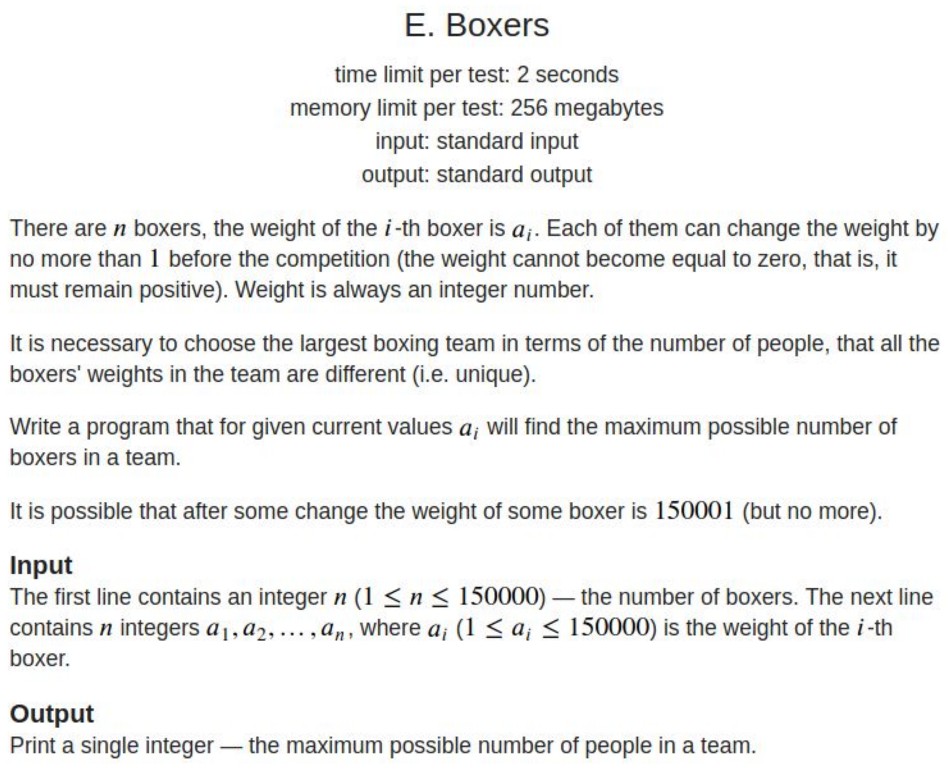
# STANDARD TEMPLATE LIBRARY ( STL )

## Iterator

```
sort(iterator_inicio, iterator_fim);

sort(iterator_inicio, iterator_fim, funcao);

unique(iterator_inicio, iterator_fim);

reverse(iterator_inicio, iterator_fim);
```

# STANDARD TEMPLATE LIBRARY ( STL )

http://www.cplusplus.com/reference/



# E. Boxers

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are $n$ boxers, the weight of the $i$-th boxer is $a_i$. Each of them can change the weight by no more than $1$ before the competition (the weight cannot become equal to zero, that is, it must remain positive). Weight is always an integer number.

It is necessary to choose the largest boxing team in terms of the number of people, that all the boxers' weights in the team are different (i.e. unique).

Write a program that for given current values $a_i$ will find the maximum possible number of boxers in a team.

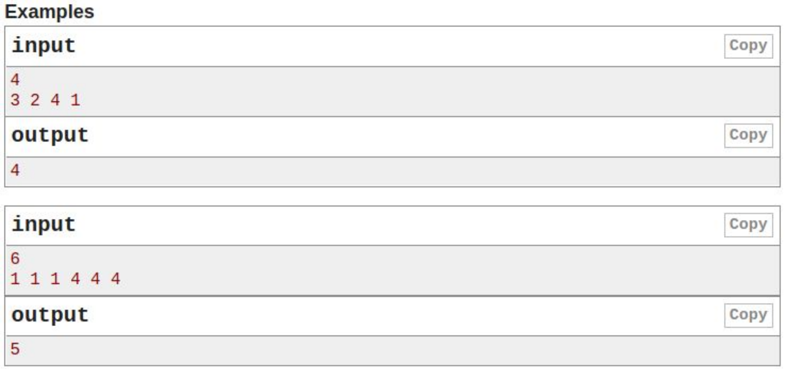
It is possible that after some change the weight of some boxer is $150001$ (but no more).

## Input

The first line contains an integer $n$ ($1 \le n \le 150000$) — the number of boxers. The next line contains $n$ integers $a_1, a_2, \dots, a_n$, where $a_i$ ($1 \le a_i \le 150000$) is the weight of the $i$-th boxer.

## Output

Print a single integer — the maximum possible number of people in a team.



### Examples

input Copy

```
4
3 2 4 1
```

output Copy

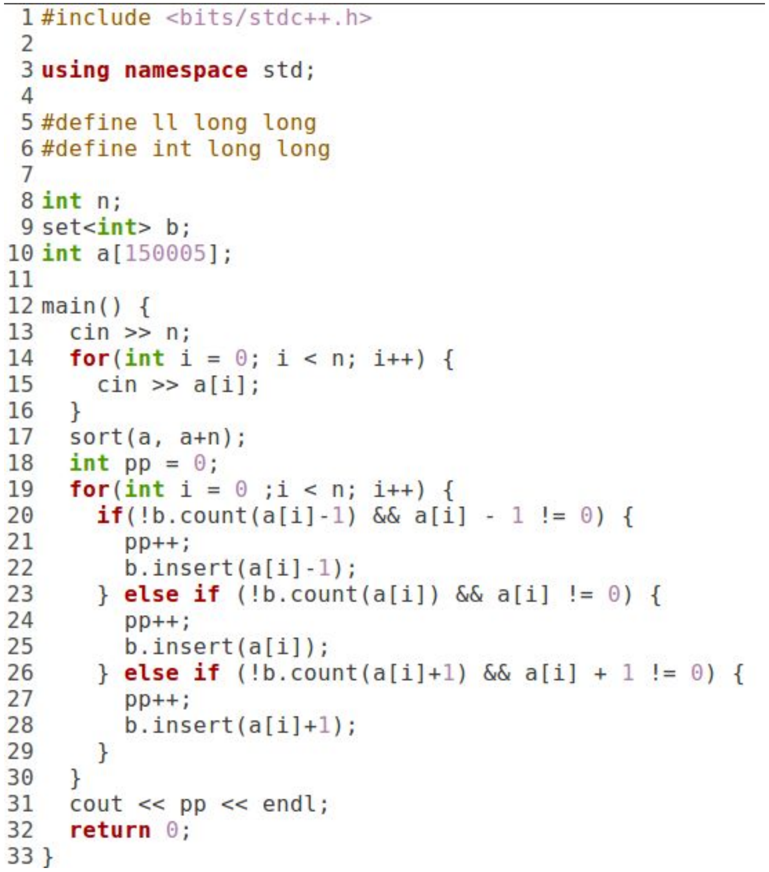
```
4
```

input Copy

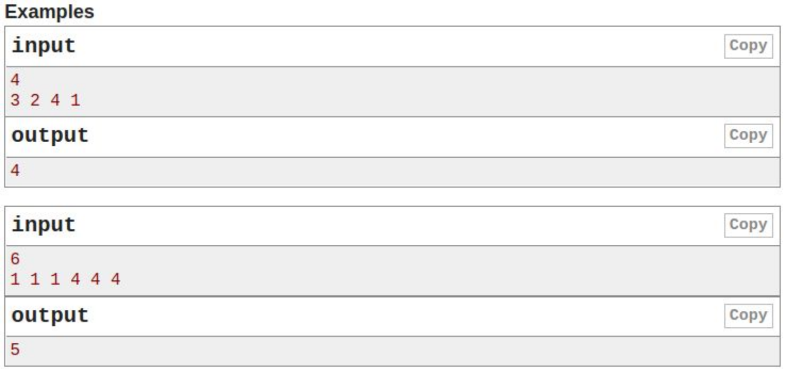
```
6
1 1 1 4 4 4
```

output Copy

```
5
```



```cpp
#include <bits/stdc++.h>

using namespace std;

#define ll long long
#define int long long

int n;
set<int> b;
int a[150005];

main() {
  cin >> n;
  for(int i = 0; i < n; i++) {
    cin >> a[i];
  }
  sort(a, a+n);
  int pp = 0;
  for(int i = 0 ;i < n; i++) {
    if(!b.count(a[i]-1) && a[i] - 1 != 0) {
      pp++;
      b.insert(a[i]-1);
    } else if (!b.count(a[i]) && a[i] != 0) {
      pp++;
      b.insert(a[i]);
    } else if (!b.count(a[i]+1) && a[i] + 1 != 0) {
      pp++;
      b.insert(a[i]+1);
    }
  }
  cout << pp << endl;
  return 0;
}
```



**Examples**

**input** Copy

```
4
3 2 4 1
```

**output** Copy

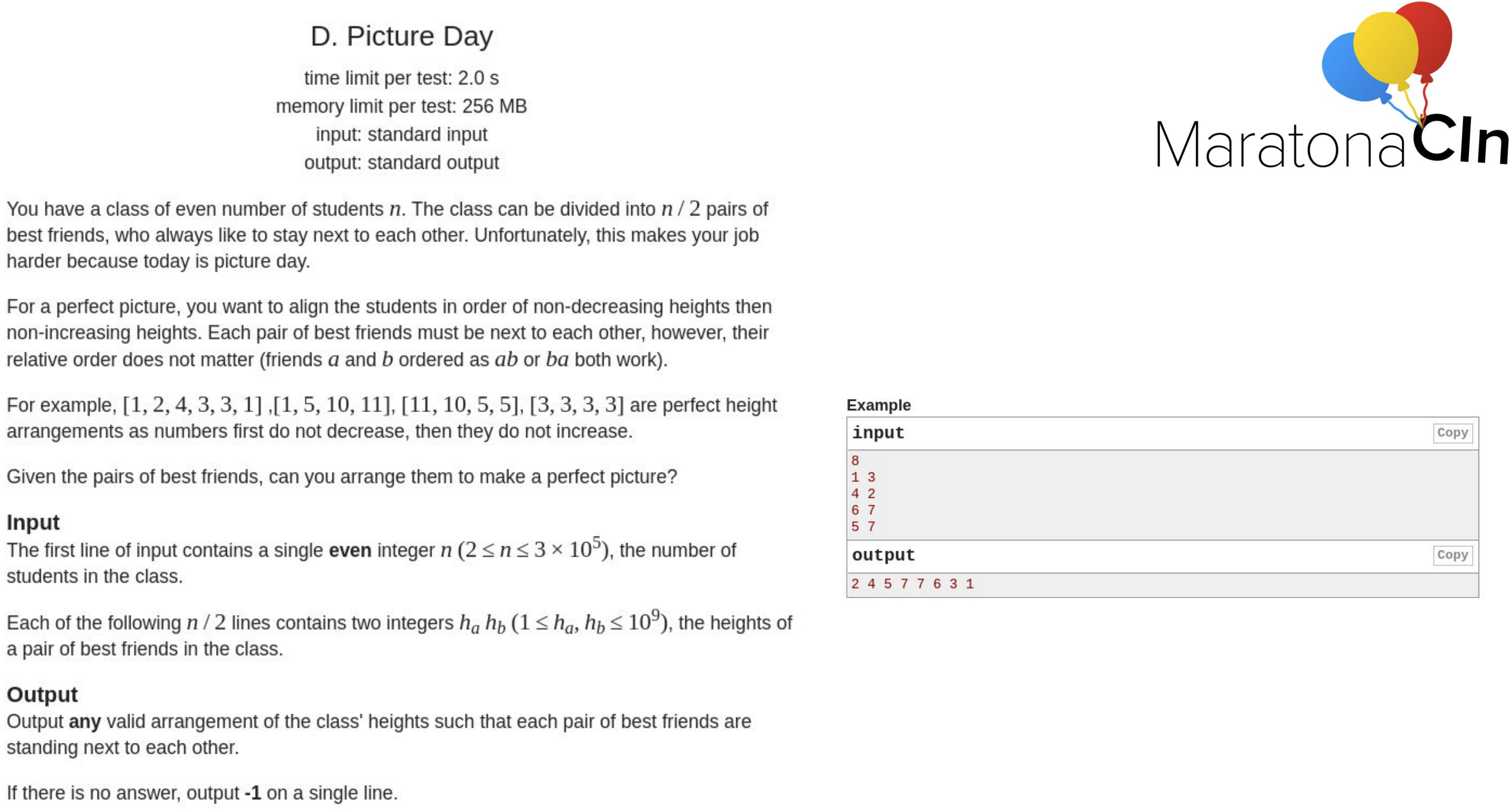
```
4
```

**input** Copy

```
6
1 1 1 4 4 4
```

**output** Copy

```
5
```



# D. Picture Day

time limit per test: 2.0 s
memory limit per test: 256 MB
input: standard input
output: standard output

You have a class of even number of students $n$. The class can be divided into $n / 2$ pairs of best friends, who always like to stay next to each other. Unfortunately, this makes your job harder because today is picture day.

For a perfect picture, you want to align the students in order of non-decreasing heights then non-increasing heights. Each pair of best friends must be next to each other, however, their relative order does not matter (friends $a$ and $b$ ordered as $ab$ or $ba$ both work).

For example, $[1, 2, 4, 3, 3, 1]$ ,$[1, 5, 10, 11]$, $[11, 10, 5, 5]$, $[3, 3, 3, 3]$ are perfect height arrangements as numbers first do not decrease, then they do not increase.

Given the pairs of best friends, can you arrange them to make a perfect picture?

## Input

The first line of input contains a single **even** integer $n$ $(2 \le n \le 3 \times 10^5)$, the number of students in the class.

Each of the following $n / 2$ lines contains two integers $h_a$ $h_b$ $(1 \le h_a, h_b \le 10^9)$, the heights of a pair of best friends in the class.

## Output

Output **any** valid arrangement of the class' heights such that each pair of best friends are standing next to each other.

If there is no answer, output **-1** on a single line.

### Example

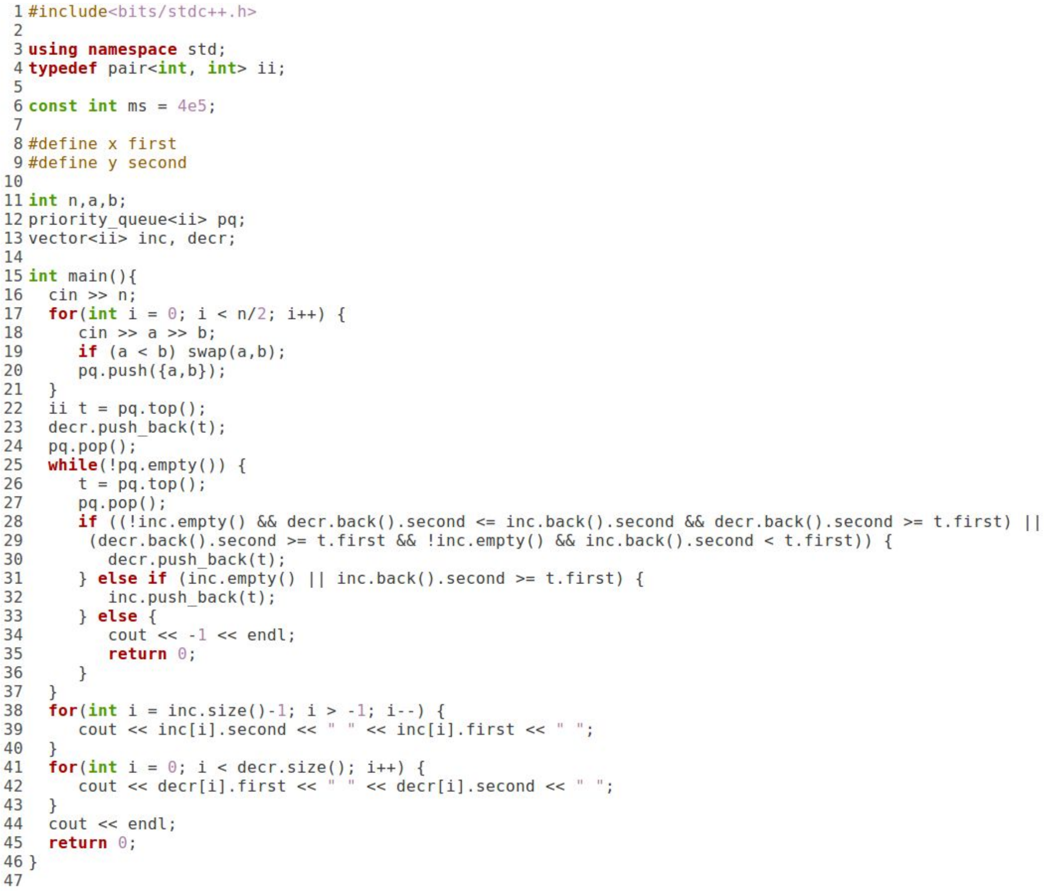
input Copy

```
8
1 3
4 2
6 7
5 7
```

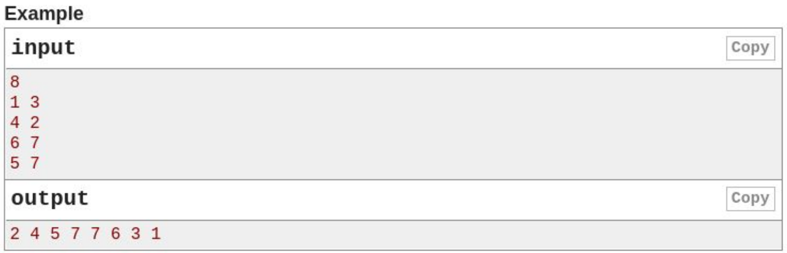
output Copy

```
2 4 5 7 7 6 3 1
```

MaratonaCIn



```cpp
#include<bits/stdc++.h>

using namespace std;
typedef pair<int, int> ii;

const int ms = 4e5;

#define x first
#define y second

int n,a,b;
priority_queue<ii> pq;
vector<ii> inc, decr;

int main(){
  cin >> n;
  for(int i = 0; i < n/2; i++) {
     cin >> a >> b;
     if (a < b) swap(a,b);
     pq.push({a,b});
  }
  ii t = pq.top();
  decr.push_back(t);
  pq.pop();
  while(!pq.empty()) {
     t = pq.top();
     pq.pop();
     if ((!inc.empty() && decr.back().second <= inc.back().second && decr.back().second >= t.first) ||
      (decr.back().second >= t.first && !inc.empty() && inc.back().second < t.first)) {
        decr.push_back(t);
     } else if (inc.empty() || inc.back().second >= t.first) {
        inc.push_back(t);
     } else {
        cout << -1 << endl;
        return 0;
     }
  }
  for(int i = inc.size()-1; i > -1; i--) {
     cout << inc[i].second << " " << inc[i].first << " ";
  }
  for(int i = 0; i < decr.size(); i++) {
     cout << decr[i].first << " " << decr[i].second << " ";
  }
  cout << endl;
  return 0;
}
```



**Example**

**input**

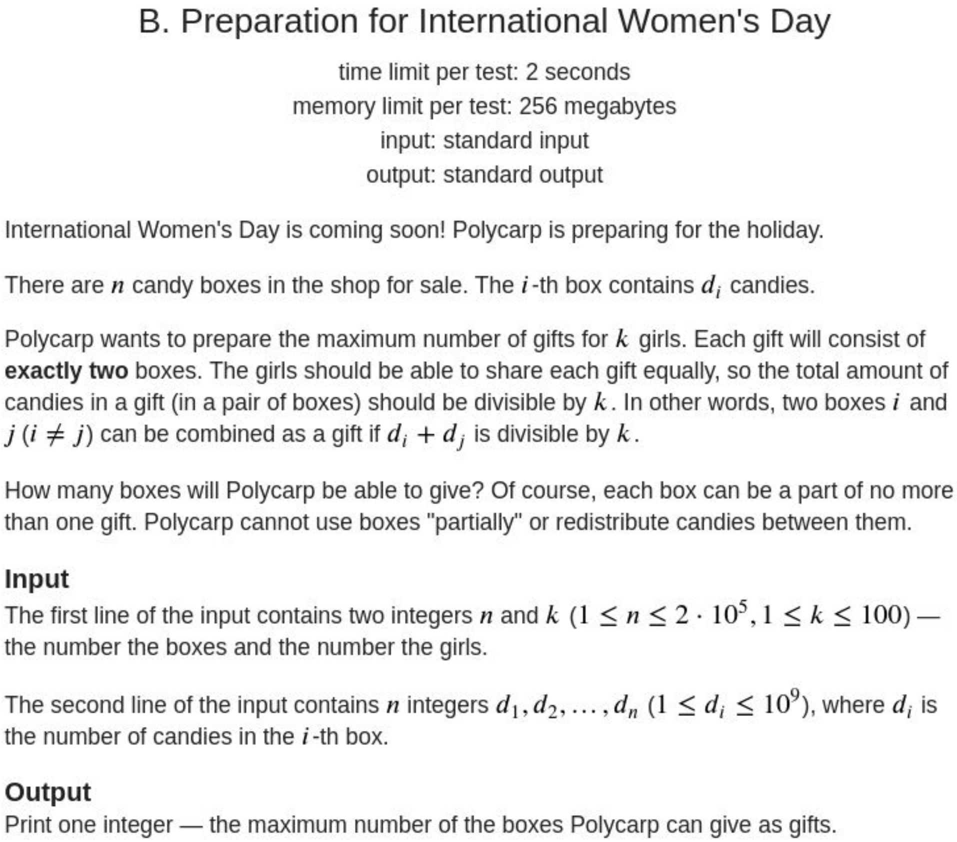
Copy

```
8
1 3
4 2
6 7
5 7
```

**output**

Copy

```
2 4 5 7 7 6 3 1
```



# B. Preparation for International Women's Day

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

International Women's Day is coming soon! Polycarp is preparing for the holiday.

There are $n$ candy boxes in the shop for sale. The $i$-th box contains $d_i$ candies.

Polycarp wants to prepare the maximum number of gifts for $k$ girls. Each gift will consist of **exactly two** boxes. The girls should be able to share each gift equally, so the total amount of candies in a gift (in a pair of boxes) should be divisible by $k$. In other words, two boxes $i$ and $j$ $(i \neq j)$ can be combined as a gift if $d_i + d_j$ is divisible by $k$.

How many boxes will Polycarp be able to give? Of course, each box can be a part of no more than one gift. Polycarp cannot use boxes "partially" or redistribute candies between them.

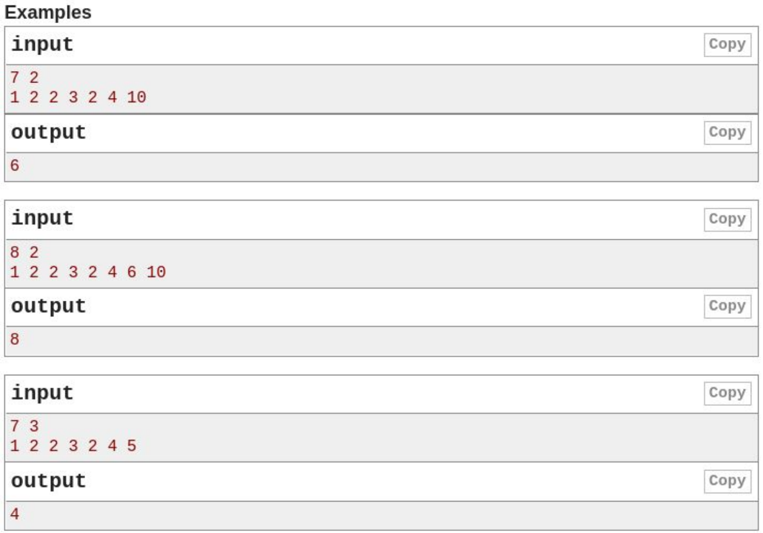
## Input

The first line of the input contains two integers $n$ and $k$ $(1 \le n \le 2 \cdot 10^5, 1 \le k \le 100)$ — the number the boxes and the number the girls.

The second line of the input contains $n$ integers $d_1, d_2, \dots, d_n$ $(1 \le d_i \le 10^9)$, where $d_i$ is the number of candies in the $i$-th box.

## Output

Print one integer — the maximum number of the boxes Polycarp can give as gifts.



Examples

input

```
7 2
1 2 2 3 2 4 10
```

output

```
6
```

input

```
8 2
1 2 2 3 2 4 6 10
```

output

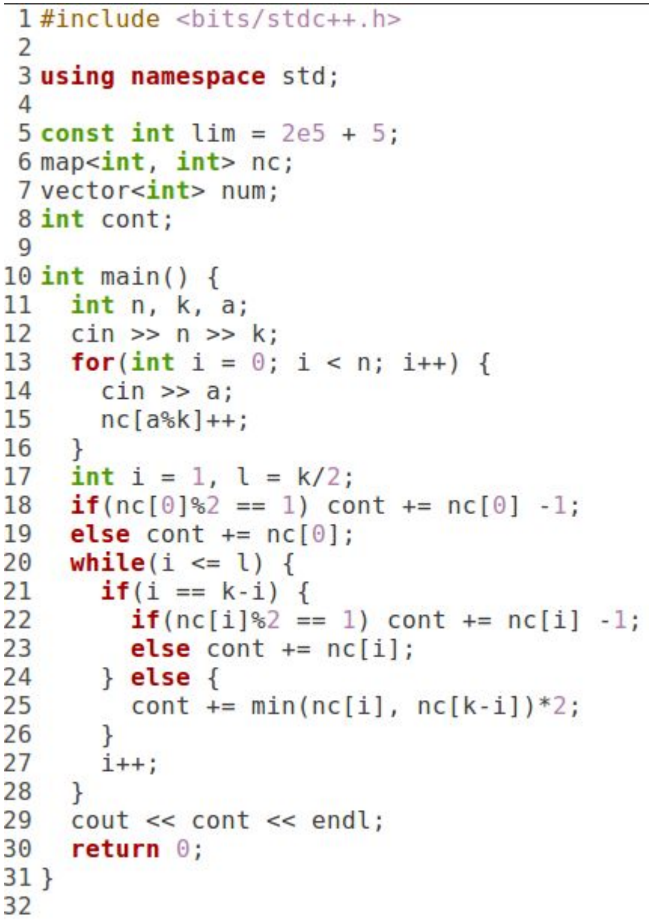
```
8
```

input

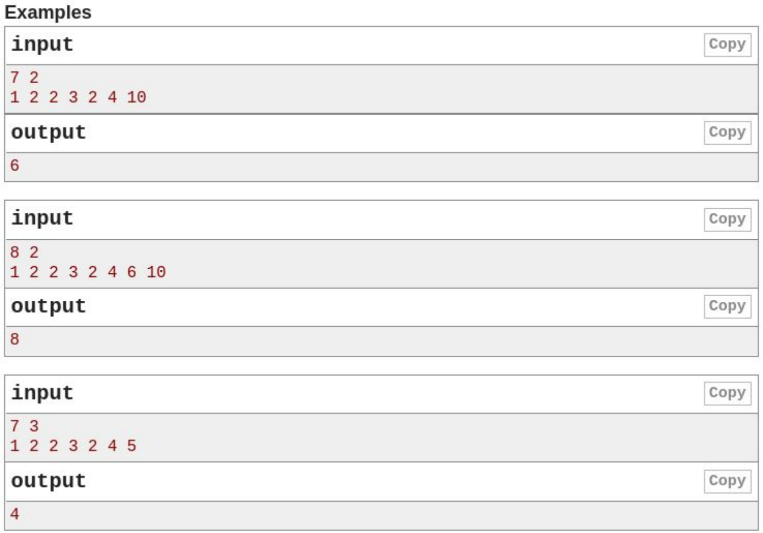
```
7 3
1 2 2 3 2 4 5
```

output

```
4
```



```cpp
#include <bits/stdc++.h>

using namespace std;

const int lim = 2e5 + 5;
map<int, int> nc;
vector<int> num;
int cont;

int main() {
  int n, k, a;
  cin >> n >> k;
  for(int i = 0; i < n; i++) {
    cin >> a;
    nc[a%k]++;
  }
  int i = 1, l = k/2;
  if(nc[0]%2 == 1) cont += nc[0] -1;
  else cont += nc[0];
  while(i <= l) {
    if(i == k-i) {
      if(nc[i]%2 == 1) cont += nc[i] -1;
      else cont += nc[i];
    } else {
      cont += min(nc[i], nc[k-i])*2;
    }
    i++;
  }
  cout << cont << endl;
  return 0;
}

```



Examples

input Copy

```
7 2
1 2 2 3 2 4 10
```

output Copy

```
6
```

input Copy

```
8 2
1 2 2 3 2 4 6 10
```

output Copy

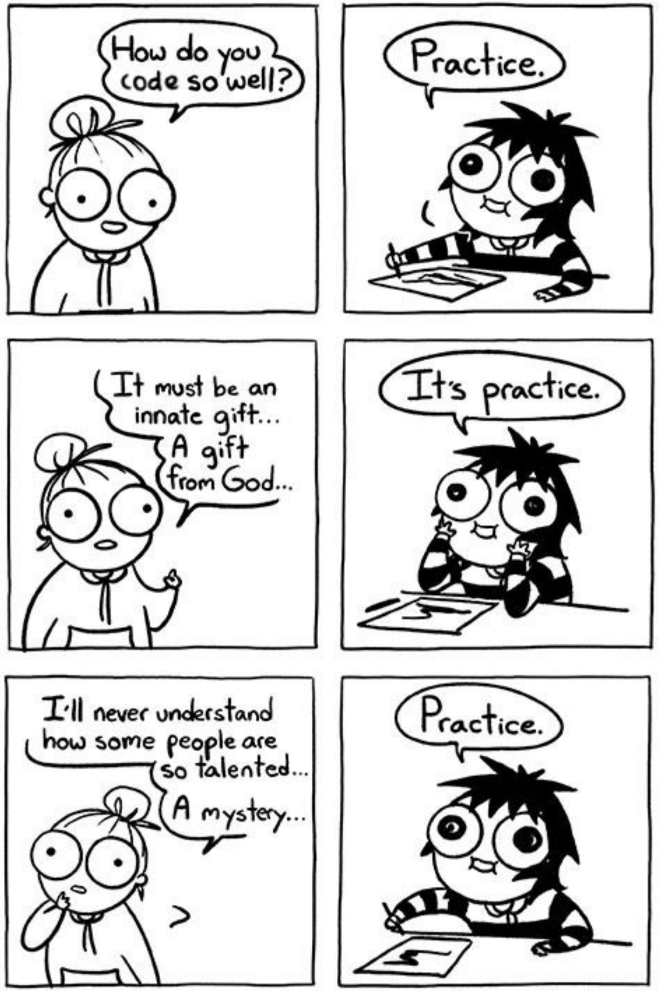
```
8
```

input Copy

```
7 3
1 2 2 3 2 4 5
```

output Copy

```
4
```


How do you code so well?
Practice.
It must be an innate gift... A gift from God...
It's practice.
I'll never understand how some people are so talented... A mystery...
Practice.