ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Spring 2013

PROGRAMMING LANGUAGES I

Exercise 1

Deadline of electronic submission: 1/5/2013, 23:59:59

The big fish eats the small.. (0.25+0.25 = 0.5 points)

In some parallel universe there is a planet that resembles earth. In a continent of this planet there exist **N** countries that, some years ago, decided to come together and form an economic union. In that union each country is characterized by its (economic) power which, for simplicity, we represent as a positive integer.

Initially there was peace and quiet in the union. However, as we all know, the big fish eat the small! In the parallel universe we study, which bears no relation with the reality we happily live in, this means that a country with size **A** sooner or later will destroy financially all other countries that have size (strictly) less than **A** / 2. In other words, the small countries, sooner or later, will be forced to declare bankruptcy and leave the union. So, without any intervention, after some years the union will end up with considerably less members!

The central government of the union (yes, governments exist everywhere!) has foreseen this situation and is ready to do what it takes in order for the union to remain with as many countries as possible. In particular, the government can remove from the union whichever countries it sees fit, in order to prevent the destruction and removal of other countries from the union. However, removing a country is a process that has a cost for the government (economically and politically) so it is essential that the government forces the removal of as few countries as possible.

What this first part of this exercise asks is to write two programs, one in C and one in ML, that take the number of countries and their sizes as input and return two numbers as output: (1) the number of countries in the optimal steady state of the union, and (2) the minimum number of countries that the government needs to remove from the union, in order for the union to reach this optimal steady state.

The input is to be read from a file, as shown in the examples that follow. The first line contains a natural number **N**, the number of countries. The second line contains **N** positive natural numbers $x_1, x_2, \ldots, x_N$ that represent the sizes of countries, separated pairwise by a single space.

Constraints: $1 \le \mathbf{N} \le 10^6$, $1 \le x_i \le 10^9$, execution time limit: 10 seconds, memory limit: 256 MB.

Below we show some example calls to the two programs in C and in ML.

In C, MLton, or in OCaml

```
> ./countries countries1.txt
5 1
> ./countries countries2.txt
1 0
> ./countries countries3.txt
6 0
```

In SML/NJ

```
- countries "countries1.txt";
val it = (5, 1) : int * int
- countries "countries2.txt";
val it = (1, 0) : int * int
- countries "countries3.txt";
val it = (6, 0) : int * int
```

Where the three files with the countries are the following ones (**cat** is a command of Unix):

```
> cat countries1.txt
7
5 9 8 4 2 4 6
> cat countries2.txt
3
1 5 11
> cat countries3.txt
6
5 8 7 7 4 6
```

In the first example, it suffices to remove a country with size 9. Then, after the dust settles, the union will consist of five countries with sizes 5, 8, 4, 4, and 6 — the country with size 2 will be destroyed anyway. There is no solution with more than five countries in the final steady state.

In the second example, the optimal state of the union will necessarily consist of only one country. So, there is no point for the government to force the removal of any country; the optimal strategy is to let the big country destroy the smaller ones.

In the third example, all countries can peacefully coexist without the need to remove any country.

Extraterrestrial basket… (0.25+0.25 = 0.5 points)

In the same parallel universe, in another planet that does not resemble earth there exist **K** kinds of creatures. In this planet they play some sport that resembles basket but has slightly different rules. Each team consists of **K** players that must contain one creature from each kind. In addition, for reasons that there is no need to explain now, it is not important to have tall players in the team but it's important for all team players to have approximately the same height.

A village mayor has decided to form a good such team and tries to find among the inhabitants of the village **K** creatures, one from each kind, such that the height difference of the tallest from the shortest is the smallest possible. The creatures of the village are **N** in total and for each creature we know their kind and height.

What this part of the exercise asks is to write two programs (one in C and one in ML) that take as input this information about the population of the village and return as output a number: the minimum height distance between the tallest and the shortest player that the best possible team that can be formed by the village inhabitants.

The input is to be read from a file, as shown in the examples that follow. The first line of the file contains two natural numbers **N** and **K**, the number of inhabitants and the number of creature kinds. Creature kinds are numbered from 1 to **K**. Each of the following **N** lines corresponds to an inhabitant of the village and consists of two natural numbers separated by a space: the kind $\mathbf{E_i}$ that the creature belongs to and its height $\mathbf{Y_i}$. You can take for granted that in all villages there exists at least one inhabitant of each creature kind, so a team can always be formed.

Constraints: $1 \le \mathbf{N} \le 10^6$, $1 \le \mathbf{K} \le 10^3$, $1 \le \mathbf{Y_i} \le 10^9$, execution time limit: 10 seconds, memory limit: 256 MB.

Below we show some possible calls to these programs in C and in ML

In C, MLton, or in OCaml

```
> ./basket basket1.txt
35
> ./basket basket2.txt
0
> ./basket basket3.txt
3
```

In SML/NJ

```
- basket "basket1.txt";
val it = 35 : int
- basket "basket2.txt";
val it = 0 : int
- basket "basket3.txt";
val it = 3 : int
```

Where the three files with the population of the villages are as follows:

```
> cat basket1.txt
5 5
2 188
1 174
5 209
3 199
4 193
```

```
> cat basket2.txt
10 5
5 89
4 96
3 37
1 89
5 91
5 94
2 89
4 89
3 62
3 89
```

```
> cat basket3.txt
8 2
1 44
1 53
2 40
2 41
2 66
1 77
1 37
1 71
```

In the first example, the mayor has no choice: there is only one team that can be formed. The height difference of the tallest from the shortest player is **`209-174=35`**.

In the second example, the mayor can select five players, one of each creature kind, all with the same height (89). So their height difference is **`0`**.

In the third example, the best team consists of the player with height **`40`** and that with height **`37`**, with height difference **`3`**.

More information on the exercises

- You can work in pairs (form teams of two students) if you want to.
- It's not permitted to share your programs with other students, to post them in places that they can be found (e.g. on the internet, in blogs, discussion fora, etc.). In case of considerable similarities between programs of separate teams, both teams will receive zero points, independently of which team got inspired from the other one. On the other hand, it is permitted to use auxiliary code from the internet (e.g. the implementation of a sorting algorithm, some data structure implementation, etc.) provided you clearly state (in comments) the source of each piece of code you use.
- Programs in C must be in one file and able to compile without warnings when using gcc (version 4.4.5) with a command of the form (for the first program):

```
gcc -Wall -O3 -o countries yourfile.c
```

- Programs in ML must also be in one file and work with SML/NJ v110.74, or MLton 20100608, or Objective Caml version 3.11.2. The submission system allows you to select between these three ML dialects.
- You can submit your programs using your moodle account (**http://moodle.softlab.ntua.gr**) using the site **http://eager.softlab.ntua.gr/plgrader**. Your programs should read the input as shown in the examples and should not contain any other output because they will not be accepted by the submission site. The submission site also contains a bigger test case that shows you the performance of your program on the machine that will be used to grade your submissions.