Problem A: Ordering supermarket queues

UCL Algorithm Contest

Round 2 - 2014

A big supermarket chain has received several complaints from their customers saying that the waiting time in queues is too long. As in any supermarket, the queues are ordered by arrival time of their customers. To overcome this problem they want to implement a new system to order people in the queues. Their objective is to **minimize the average waiting time of the customers in the queue.**

Task

Given the number of persons n and the time each persons will take, t_1, t_2, ..., t_n, compute the minimum average waiting time of the persons in an optimal ordering of the queue.

Example

Suppose $n = 2$ and $t_1 = 1$, $t_2 = 2$. There are two possible ways to order those two persons: 1, 2 and 2, 1. In the first way has average time $(0 + 1) / 2 = 1 / 2$ and the second way has average time $(0 + 2) / 2 = 2 / 2 = 1 / 1$. Hence the answer is $1/2$. Note that the first person does not have to wait to be served.

Input Specification

The input consists of two lines. The first contains one integer n giving the number of persons in the queue. The second line contains n integers separated by spaces, t_1, t_2, ..., t_n that correspond to the time each persons takes.

Constraints

- $2 \leq n \leq 10^5$

- $1 \leq t_i \leq 1000$

Warning: use long variables!

Output Specification

The output is the average waiting time represented as a **reduced fraction**. You have to output one line with the format "a[space]/[space]b" (without the quotes) where *a / b* equals the minimal average waiting time and *gcd(a, b) = 1*. You can compute the *gcd* of two integers with:

```
gcd(a, b)
  if(b == 0) return a
  else return gcd(b, a % b)
```

The reduced form of a fraction a / b is (a / gcd(a, b)) / (b / gcd(a, b)).

Sample Input 1

```
2
1 2
```

Sample Output 1

```
1 / 2
```

Sample Input 2

```
4
7 4 9 3
```

Sample Output 2

```
6 / 1
```

Sample Input 3

```
100000
1000 1000 1000 ... 1000         (100000 times)
```

Sample Output 3

```
49999500 / 1
```

Problem B: Playing with marbles

UCL Algorithm Contest

Round 2 - 2014

Alice and Bob are playing a game. To start the game they place n marbles on a table and agree on a set of numbers k_1, k_2, ..., k_t. Then in turns they choose one of the numbers k_i and remove that number of marbles from the table. Of course a player is only allowed to choose a number that is smaller or equal to the current number of marbles on the table. If a player cannot play he looses.

Alice is always the first to play and she would like you to help her determine if she will win of loose.

Example

Suppose $n = 6$ and that there are two possible moves: removing either 2 or 3 marbles. If Alice removes 2 marbles then there will remain 4 marbles on the table. Then if Bob removes 3 marbles only 1 stone remains and Alice looses because she cannot play. On the other hand, if Alices removes 3 marbles then there will remain 3 marbles on the table and any move that Bob make will make Alice loose.

Task

Given n and the possible moves k_1, k_2, ..., k_t, if both players play perfectly, determine who wins the game. Recall that Alice always plays first.

Input Specification

The input consists of two lines. The first contains two integers n and t separated by a single space. The second line contains t integers separated by spaces, k_1, k_2, ..., k_t that correspond to the moves they agreed on.

Constraints

- $1 \leq n \leq 10^9$
- $1 \leq t \leq 10$

- $1 \leq k_i \leq n$
- $max(k_i) \leq 20$

Output Specification

The output consists of a single line with "Alice" if Alice wins and "Bob" if Bob wins (without the quotes).

Sample Input 1

```
6 2
2 3
```

Sample Output 1

```
Bob
```

Sample Input 2

```
16 4
1 2 3 4
```

Sample Output 2

```
Alice
```

Problem C: Pile shuffle

UCL Algorithm Contest

Round 2 - 2014

Alice has a deck of *n* cards numbered from 1 to *n*. The cards are initialy in the order 1, 2, 3, 4, ..., *n*. She wants to shuffle the cards and to do so she will choose some number *p* and make *p* piles of cards. She will put the first card on the first pile, the second card on the second pile, ..., the *p*-th card on the *p*-th pile and then she goes back to the first pile and continues. After all the cards are placed she will take the first pile and put it over the second, then take the resulting pile and put it over the third and so on, until she has only one pile again.

Example

If *n = 8* and *p = 3* the piles will be as follows:

```
     7  8       top card
     4  5  6
     1  2  3    bottom card
------------
pile 1  2  3
```

Then she takes the first pile at puts and over the second pile:

```
        7       top card
        4
        1
        8
        5  6
        2  3    bottom card
------------
pile 1  2  3
```

Finaly she takes the resulting pile and puts it over the third pile getting only one pile with the cards in the following order:

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

Task

Given n and p output the result of one iteration of Alice's pile shuffle.

Input Specification

A single line with two integers n and p separated by a single space. The first is the number of cards and the second is the number of piles.

Constraints

- $1 \leq n \leq 50000$
- $1 \leq p \leq n$

Output Specification

A single line with the number of the cards after one pile shuffle (from top to bottom) separated by a single space as in the example above.

Sample Input 1

```
10 1
```

Sample Output 1

```
10 9 8 7 6 5 4 3 2 1
```

Sample Input 2

```
10 10
```

Sample Output 2

```
1 2 3 4 5 6 7 8 9 10
```

Sample Input 3

```
14 5
```

Sample Output 3

```
11 6 1 12 7 2 13 8 3 14 9 4 10 5
```

Problem D: Assembling the pieces

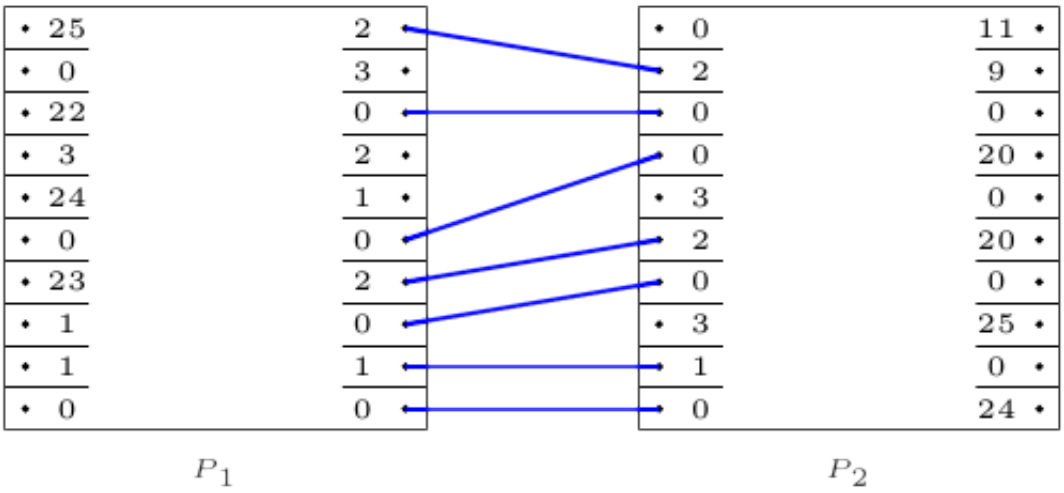
UCL Algorithm Contest

Round 2 - 2014

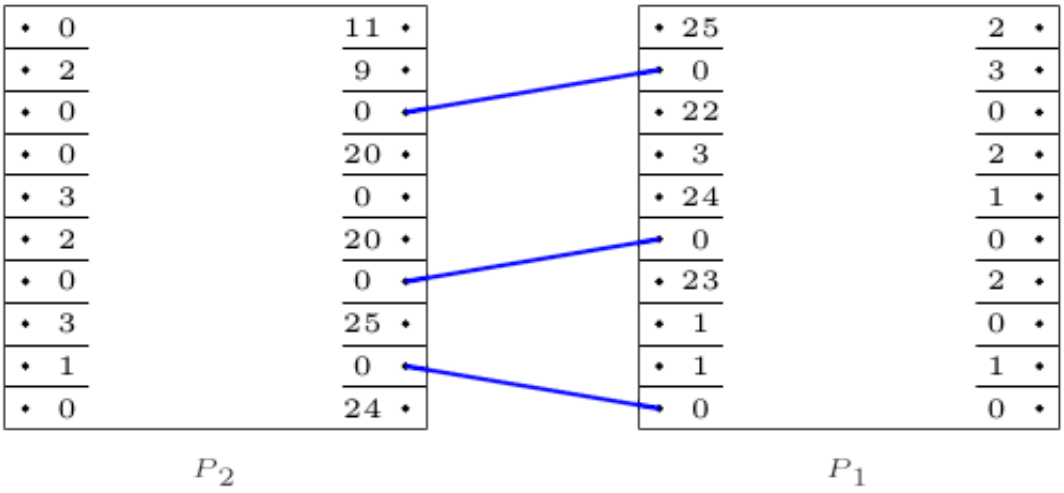
You are given square pieces each with 10 connectors on their left and right sides. Each connector has a label that is represented by an integer. Two pieces P_1 and P_2 can be assembled together if the energy between the right side connectors of P_1 and the left side connectors of P_2 is at least 5 (half of the connectors). The energy between two sides is measured as follows. Let R be the sequence of labels that represent the right side of P_1 and L be the sequence of labels that represent the left side of P_2. If $R_i = L_j$ then an enery link can be created between those connectors. The energy between R and L is defined as the maximum number of non-intersecting energy links that can exist between R and L.

Example

The following image shows two pieces P_1 and P_2. We can create a chain P_1-P_2 because the energy between the right side of P_1 and the left side of P_2 is 7 >= 5.



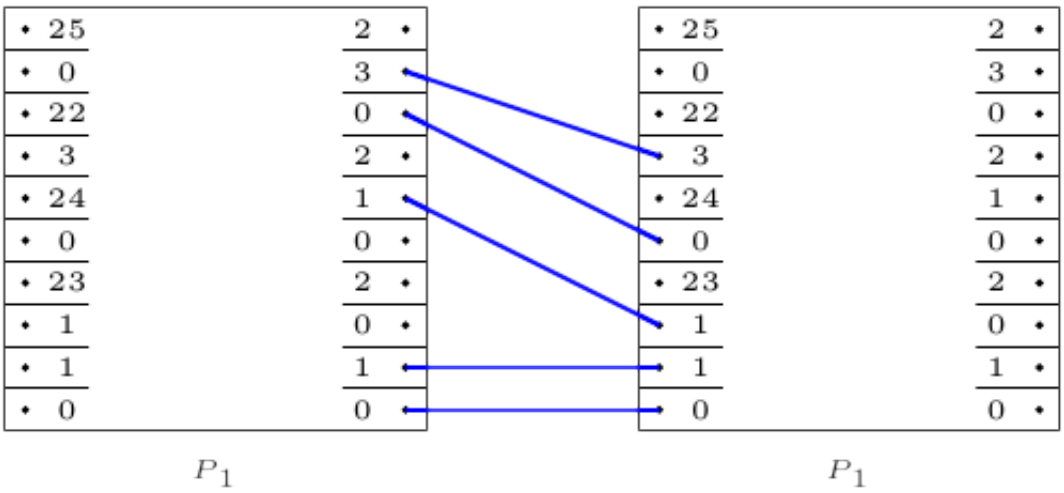
However we cannot create a chain P_2-P_1 because the energy between the right side of P_2 and the left side of P_1 is only 3 < 5.



We cannot add the link between connectors 25 since it would intersect the other links.

Note that there can be more that one way to create the maximum number of non-intersecting links but we only care about the number of links on those solutions.

Also, in this case the right side of P_1 can be connected to its left side so we can create a chain P_1-P_1 if we have two pieces of type P_1 available.



Task

You will be given the description of several types of pieces. Supposing that you have an infinite amout of pieces of each type you are asked to determine if it is possible to create an infinite chain. In the case it is possible you have to compute the minimum number of different types of pieces that are needed to do so.

Input Specification

The first line contains a single integer *n* representing the number of differents types of pieces that are available. Then follow *2n* lines each with 10 integers giving the labels of the left side connectors and the

labels of the right side connectors, respectively.

Constraints

- $1 \leq n \leq 100$
- The labels are integers $< 2^{31}$

Output Specification

A single line with either the minimum number of different types of pieces that are needed to build an infinite chain or "Impossible" if there is no way to build an infinite chain.

Sample Input 1

```
2
25 0 22 3 24 0 23 1 1 0
2 3 0 2 1 0 2 0 1 0
0 2 0 0 3 2 0 3 1 0
11 9 0 20 0 20 0 25 0 24
```

Sample Output 1

```
1
```

Sample Input 2

```
3
0 0 7 8 4 1 8 3 7 4
0 4 6 7 0 3 8 9 3 1
4 1 1 0 4 7 6 1 7 2
7 6 4 4 5 4 7 9 9 3
6 5 4 2 0 2 3 8 9 8
0 0 4 7 2 2 6 0 7 4
```

Sample Output 2

```
2
```

Sample Input 3

```
3
52 70 94 90 4 53 68 11 85 30
54 33 57 13 0 51 79 16 79 50
31 4 13 82 63 79 93 32 66 25
82 3 85 11 43 5 59 1 81 70
19 52 18 65 21 69 69 70 62 65
24 19 41 38 57 19 78 91 85 53
```

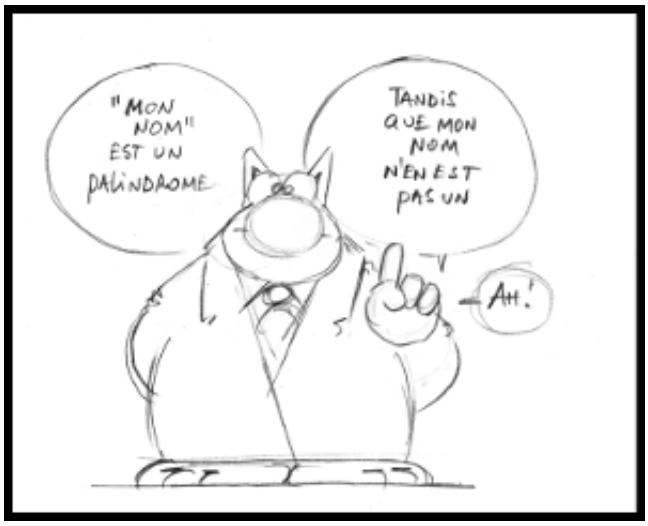
Sample Output 3

```
Impossible
```

Problem E: Palindromic anagrams

UCL Algorithm Contest

Round 2 - 2014



A palindrome is a string that reads the same from the left and from the right. Here are some examples: KAYAK, RADAR, MADAM, LEVEL, ...

A string x is anagram of another string y if we can rearrange the characters of x and obtain the string y. For example string DOCTORWHO is an anagram of the string TORCHWOOD.

Task

Given a string s we want to delete as few character as possible from it so that the resulting string is an anagram to some palindrome. Your task is to find what is the minimum number of characters that have to be deleted.

Input Specification

A single line with a string s of the alphabet {A, B, C, D, E, F, G, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}.

Constraints

Let $|s|$ denote the length of s.

- $1 \leq |s| \leq 100000$

Output Specification

A single line with the minimum number of characters that have to be deleted.

Sample Input 1

```
YUNOAC
```

Sample Output 1

```
5
```

Sample Input 2

```
ACCEPTED
```

Sample Output 2

```
3
```