Kotlin Heroes: Practice 7

A. A+B (Trial Problem)

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given two integers a and b. Print $a + b$.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. Then t test cases follow.

Each test case is given as a line of two integers a and b ($-1000 \le a, b \le 1000$).

Output

Print t integers — the required numbers $a + b$.

Example

input

```
4
1 5
314 15
-99 99
123 987
```

output

```
6
329
0
1110
```

B. Frog Jumping

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

A frog is currently at the point 0 on a coordinate axis Ox. It jumps by the following algorithm: the first jump is a units to the right, the second jump is b units to the left, the third jump is a units to the right, the fourth jump is b units to the left, and so on.

Formally:

- if the frog has jumped an even number of times (before the current jump), it jumps from its current position x to position $x + a$;
- otherwise it jumps from its current position x to position $x - b$.

Your task is to calculate the position of the frog after k jumps.

But... One more thing. You are watching t different frogs so you have to answer t independent queries.

Input

The first line of the input contains one integer t ($1 \le t \le 1000$) — the number of queries.

Each of the next t lines contain queries (one query per line).

The query is described as three space-separated integers a, b, k ($1 \le a, b, k \le 10^9$) — the lengths of two types of jumps and the number of jumps, respectively.

Output

Print t integers. The i-th integer should be the answer for the i-th query.

Example

input

```
6
5 2 3
100 1 4
1 10 5
1000000000 1 6
1 1 1000000000
1 1 999999999
```

output

```
8
198
-17
2999999997
0
1
```

Note

In the first query frog jumps 5 to the right, 2 to the left and 5 to the right so the answer is $5 - 2 + 5 = 8$.

In the second query frog jumps 100 to the right, 1 to the left, 100 to the right and 1 to the left so the answer is $100 - 1 + 100 - 1 = 198$.

In the third query the answer is $1 - 10 + 1 - 10 + 1 = -17$.

In the fourth query the answer is $10^9 - 1 + 10^9 - 1 + 10^9 - 1 = 2999999997$.

In the fifth query all frog's jumps are neutralized by each other so the answer is 0.

The sixth query is the same as the fifth but without the last jump so the answer is 1.

C. Uniform String

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given two integers n and k.

Your task is to construct such a string s of length n that for each i from 1 to k there is at least one i-th letter of the Latin alphabet in this string (the first letter is 'a', the second is 'b' and so on) and there are no other letters except these. You have to **maximize the minimal frequency** of some letter (the frequency of a letter is the number of occurrences of this letter in a string). If there are several possible answers, you can print **any**.

You have to answer t **independent** queries.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of queries.

The next t lines are contain queries, one per line. The i-th line contains two integers n_i and k_i ($1 \le n_i \le 100, 1 \le k_i \le min(n_i, 26)$) — the length of the string in the i-th query and the number of characters in the i-th query.

Output

Print t lines. In the i-th line print the answer to the i-th query: **any** string s_i satisfying the conditions in the problem statement with constraints from the i-th query.

Example

input

```
3
7 3
4 4
6 2
```

output

```
cbcacab
abcd
baabab
```

Note

In the first example query the maximum possible minimal frequency is 2, it can be easily seen that the better answer doesn't exist. Other examples of correct answers: "`cbcabba`", "`ccbbaaa`" (any permutation of given answers is also correct).

In the second example query any permutation of first four letters is acceptable (the maximum minimal frequency is 1).

In the third example query any permutation of the given answer is acceptable (the maximum minimal frequency is 3).

D. Teams Forming

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

There are n students in a university. The number of students is even. The i-th student has programming skill equal to a_i.

The coach wants to form $\frac{n}{2}$ teams. Each team should consist of exactly two students, and each student should belong to exactly one team. Two students can form a team only if their skills are equal (otherwise they cannot understand each other and cannot form a team).

Students can solve problems to increase their skill. One solved problem increases the skill by one.

The coach wants to know the minimum total number of problems students should solve to form exactly $\frac{n}{2}$ teams (i.e. each pair of students should form a team). Your task is to find this number.

Input

The first line of the input contains one integer n ($2 \le n \le 100$) — the number of students. It is guaranteed that n is even.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$), where a_i is the skill of the i-th student.

Output

Print one number — the minimum total number of problems students should solve to form exactly $\frac{n}{2}$ teams.

Examples

input

```
6
5 10 2 3 14 5
```

output

```
5
```

input

```
2
1 100
```

output

```
99
```

Note

In the first example the optimal teams will be: $(3, 4)$, $(1, 6)$ and $(2, 5)$, where numbers in brackets are indices of students. Then, to form the first team the third student should solve 1 problem, to form the second team nobody needs to solve problems and to form the third team the second student should solve 4 problems so the answer is $1 + 4 = 5$.

In the second example the first student should solve 99 problems to form a team with the second one.

E. Good Array

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Let's call an array *good* if there is an element in the array that equals to the sum of all other elements. For example, the array $a = [1, 3, 3, 7]$ is good because there is the element $a_4 = 7$ which equals to the sum $1 + 3 + 3$.

You are given an array a consisting of n integers. Your task is to print all indices j of this array such that after removing the j-th element from the array it will be *good* (let's call such indices *nice*).

For example, if $a = [8, 3, 5, 2]$, the *nice* indices are 1 and 4:

- if you remove a_1, the array will look like $[3, 5, 2]$ and it is *good*;
- if you remove a_4, the array will look like $[8, 3, 5]$ and it is *good*.

You have to consider all removals **independently**, i. e. remove the element, check if the resulting array is *good*, and return the element into the array.

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of elements in the array a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$) — elements of the array a.

Output

In the first line print one integer k — the number of indices j of the array a such that after removing the j-th element from the array it will be *good* (i.e. print the number of the *nice* indices).

In the second line print k distinct integers $j_1, j_2, \dots, j_k$ in **any** order — *nice* indices of the array a.

If there are no such indices in the array a, just print 0 in the first line and leave the second line empty or do not print it at all.

Examples

input

```
5
2 5 1 2 2
```

output

```
3
4 1 5
```

input

```
4
8 3 5 2
```

output

```
2
1 4
```

input

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

output

```
0
```

Note

In the first example you can remove any element with the value 2 so the array will look like $[5, 1, 2, 2]$. The sum of this array is 10 and there is an element equals to the sum of remaining elements ($5 = 1 + 2 + 2$).

In the second example you can remove 8 so the array will look like $[3, 5, 2]$. The sum of this array is 10 and there is an element equals to the sum of remaining elements ($5 = 3 + 2$). You can also remove 2 so the array will look like $[8, 3, 5]$. The sum of this array is 16 and there is an element equals to the sum of remaining elements ($8 = 3 + 5$).

In the third example you cannot make the given array *good* by removing exactly one element.

F. Prefixes and Suffixes

time limit per test: 2.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

Ivan wants to play a game with you. He picked some string s of length n consisting only of lowercase Latin letters.

You don't know this string. Ivan has informed you about all its improper prefixes and suffixes (i.e. prefixes and suffixes of lengths from 1 to $n-1$), but he didn't tell you which strings are prefixes and which are suffixes.

Ivan wants you to guess which of the given $2n-2$ strings are prefixes of the given string and which are suffixes. It may be impossible to guess the string Ivan picked (since multiple strings may give the same set of suffixes and prefixes), but Ivan will accept your answer if there is at least one string that is consistent with it. Let the game begin!

Input

The first line of the input contains one integer number n ($2 \le n \le 100$) — the length of the guessed string s.

The next $2n-2$ lines are contain prefixes and suffixes, one per line. Each of them is the string of length from 1 to $n-1$ consisting only of lowercase Latin letters. They can be given in arbitrary order.

It is guaranteed that there are exactly 2 strings of each length from 1 to $n-1$. It is also guaranteed that these strings are prefixes and suffixes of some existing string of length n.

Output

Print one string of length $2n-2$ — the string consisting only of characters 'P' and 'S'. The number of characters 'P' should be equal to the number of characters 'S'. The i-th character of this string should be 'P' if the i-th of the input strings is the prefix and 'S' otherwise.

If there are several possible answers, you can print **any**.

Examples

input

```
5
ba
a
abab
a
aba
baba
ab
aba
```

output

```
SPPSPSPS
```

input

```
3
a
aa
aa
a
```

output

```
PPSS
```

input

```
2
a
c
```

output

```
PS
```

Note

The only string which Ivan can guess in the first example is "ababa".

The only string which Ivan can guess in the second example is "aaa". Answers "SPSP", "SSPP" and "PSPS" are also acceptable.

In the third example Ivan can guess the string "ac" or the string "ca". The answer "SP" is also acceptable.