Add

Write a program that reads two integers from standard input, calculates their sum and prints the result to standard output.

Input

Two integer values -1,000,000,000 <= v1, v2 <= 1,000,000,000 each on its own line.

Output

A single value that is the sum of v1 and v2 followed by a newline.

Example 1

```
Input:
4
9

Output:
13
```

Example 2

```
Input:
-5
5

Output:
0
```

Here are sample solutions to this problem in C, C++ and Java:

```c
#include <stdio.h>

int main()
{
    int a,b;
    scanf("%d %d",&a, &b);
    printf("%d\n",a+b);
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int a,b;
    cin >> a >> b;
    cout << a+b << endl;
    return 0;
}
```

```java
import java.util.Scanner;

public class Main
{
    public static void main(String args[])
    {
```

```
        Scanner in=new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        System.out.println(a + b);
    }
}
```

Suggestion 1: write all three solutions on your own and submit them to the autograder - don't just copy and paste.

Suggestion 2: if you are unfamiliar with one of the above languages, study its solution carefully and make sure you understand each line of the code.

Add, 2

Write a program that reads two integers from standard input, calculates their sum and prints the result to standard output.

Input

Two integer values -2,000,000,000 <= v1, v2 <= 2,000,000,000 each on its own line.

Output

A single value that is the sum of v1 and v2 followed by a newline.

Example 1

```
Input:
4
9

Output:
13
```

Example 2

```
Input:
-5
5

Output:
0
```

Fibonacci numbers

In mathematics Fibonacci numbers are the the sequence of numbers such that each value is the sum of the two preceding values starting with the first two values

fib1 = 1

fib2 = 1

(Yes, there are some definitions that start with a pair: 0, 1, but for the purpose of this problem we will use the above starting values.)

Your task is to compute the N'th Fibonacci number given the number N.

Restriction: Your algorithm has to be recursive, but it needs to be efficient to pass all the test. If it is not, the score for this problem will be set to zero regardless of the autograder score.

Input

The first and only line of input contains the number N (1 <= N <= 40).

Output

The output contains the value of the N'th Fibonacci number followed by a newline character.

Example 1

```
Input:
1

Output:
1
```

Example 2

```
Input:
4

Output:
3
```

Fibonacci numbers, 2

In mathematics Fibonacci numbers are the the sequence of numbers such that each value is the sum of the two preceding values starting with the first two values

fib1 = 0

fib2 = 1

(Yes, there are some definitions that start with a pair: 1, 1, but for the purpose of this problem we will use the above starting values.)

Your task is to compute the N'th Fibonacci number given the number N.

Input

The first and only line of input contains the number N (1 <= N <= 80).

Output

The output contains the value of the N'th Fibonacci number followed by a newline character.

Example 1

```
Input:
1

Output:
0
```

Example 2

```
Input:
4

Output:
2
```

Perfect Numbers

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3 (excluding itself), and $1 + 2 + 3 = 6$, so 6 is a perfect number. On the other hand, 8 has divisors 1, 2 and 4 (excluding itself), and $1 + 2 + 4 = 7$, so it is not a perfect number.

Your task is to write a program that determines if a number is perfect or not.

Input

A single integer $1 <= n <= 2^{60}$ on a single line.

Output

If n is perfect the program should output PERFECT. Otherwise, the program should output NOT PERFECT. In either case, the output should terminate with a newline.

Example 1

```
Input:
6

Output:
PERFECT
```

Example 2

```
Input:
8

Output:
NOT PERFECT
```

Example 3

```
Input:
14

Output:
NOT PERFECT
```

Car Value

Most buyers of new cars take a bank loan for the purchase. You probably heard about the fact that a new car loses 10% of its value the minute you drive it off the lot. So it is likely that when a buyer takes a loan for the car and then drives it off the lot, they actually owe the bank more than the car is worth.

Each month the buyer's loan payment reduces the amount owed on the car. However, each month, the car also depreciates as it gets older. Your task is to write a program that calculates the first time, measured in months, that a car buyer owes less money than a car is worth. For this problem, depreciation is specified as a percentage of the previous month's value.

For simplicity, we will assume a 0% interest loan.

The car's initial value is equal to the loan amount plus the down payment.

Figure 1: Black Man Driving Car Cartoon Vector, source: Videoplasty.com, CC BY-SA 4.0, via Wikimedia Commons

Input

Input consists of information for a single loan. Each loan consists of one line containing the duration of the loan in months, the down payment, the amount of the loan, and the number of depreciation records that follow. All values are non-negative, with loans being at most 100 months long and car values at most $75,000. Since depreciation is not constant, the varying rates are specified in a series of depreciation records. Each depreciation record consists of one line with a month number and depreciation percentage, which is more than 0 and less than 1. These are in strictly increasing order by month, starting at month 0. Month 0 is the depreciation that applies immediately after driving the car off the lot and is always present in the data. All the other percentages are the amount of depreciation at the end of the corresponding month. Not all months may be listed in the data. If a month is not listed, then the previous depreciation percentage applies.

Output

The output is the number of complete months before the borrower owes less than the car is worth followed by a newline. Note that English requires plurals (5 months) on all values other than one (1 month).

It is possible for a car's value and amount owed to be positive numbers less than $1.00. Do not round values to a whole number of cents ($7,347.635 should not be rounded to$7,347.64).

Example 1

Consider the case of borrowing $15,000 for 30 months. As the buyer drives off the lot, they still owe $15,000, but the car has dropped in value by 10% to $13,950. After 4 months, the buyer has made 4 payments, each of $500, and the car has further depreciated 3% in months 1 and 2 and 0.2% in months 3 and 4. At this time, the car is worth $13,073.10528 and the borrower only owes $13,000.00.

```
Input:
30 500.0 15000.0 3
0 .10
1 .03
3 .002

Output:
4 months
```

Example 2

```
Input:
12 500.0 9999.99 2
0 .05
2 .1

Output:
1 month
```

Example 3

```
Input:
60 2400.0 30000.0 3
0 .2
1 .05
12 .025

Output:
49 months
```

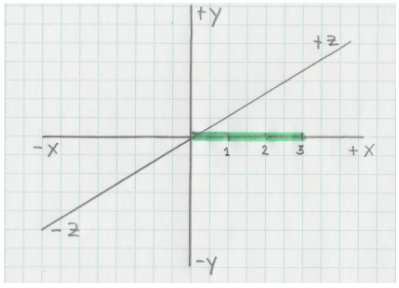
Rod Sculpture

A group of CS students decided to enter The Abstract Art contest at their university. Since none of the students have any specific art skills, they came up with an idea of programming a robot to create their art project. The robot's task is to bend a metal rod according to a specification provided to it.

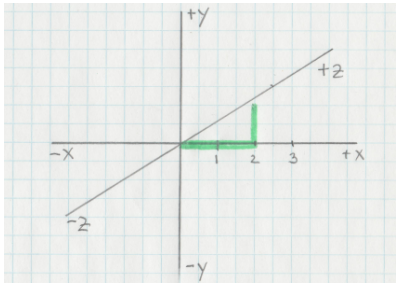
The first version of the robot is ready and it can perform a limited task of bending a metal rod by ninety degrees in different directions.

The robot views the metal rod as positioned along the positive X-axis, +x, in a three-dimensional coordinate system anchored at the origin. The rod is of length L (L is an integer in the range 2 <= L <= 100,000). This means that the rod is attached at point (0,0,0) and its free (bendable) end is at (L,0,0). The robot can bend the rod at discrete points starting from (L-1,0,0) down to (1,0,0). At each point i the robot can either: - not bend the rod at point (i,0,0) - bend the rod at point (i,0,0) at angle of 90 degrees so that the rod segment from (i,0,0) to (i+1,0,0) is parallel to the axis +y, -y, +z or -z.

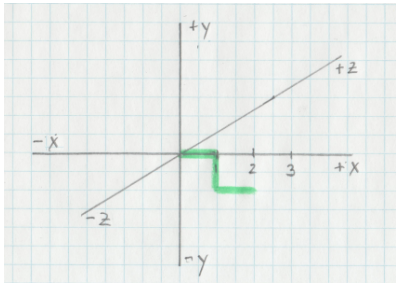
The following example shows what happens for a rod of length L=3 and the sequence of instructions `+y -y`.



(a) initial rod of length L=3



(b) after bend +y at (2,0,0)



(c) after bend -y at (1,0,0)

Your task is to determine in what direction the last segment of the rod is pointing to. In the above example, that direction is `+x`. You can assume that the rod is thin enough that it can intercept itself.

Input

The first line specifies the length of the rod, L, as an integer 2 <= L <= 100,000. The second line contains L-1 instructions for the robot separated by spaces. The j'th instruction (1 <= j <= L-1) corresponds to a bend at point (L-j, 0, 0) and must be one of the following: - `No` - do not bend the rod at point (L-j, 0, 0) - `+y` - bend the rod at point (L-j,0,0) on the positive Y-axis - `-y` - bend the rod at point (L-j,0,0) on the negative Y-axis - `+z` - bend the rod at point (L-j,0,0) on the positive Z-axis - `-z` - bend the rod at point (L-j,0,0) on the negative Z-axis

Output

The direction in which the last segment of the rod is pointing to as `+x`, `-x`, `+y`, `-y`, `+z`, or `-z`.

Example 1

```
Input:
3
+z -z

Output:
+x
```

Example 2

```
Input:
3
+z +y

Output:
+z
```

Example 3

```
Input:
5
No +z No No

Output:
+z
```