Question - 1
Adding Two Numbers

Determine the integer floor of the sum of two floating point numbers. The floor is the truncated float value, i.e. anything after the decimal point is dropped.

Example
floor(1.1 + 3.89) = floor(4.99) = 4.

Function Description
Complete the function *addNumbers* in the editor below.

addNumbers has the following parameter(s):
float a: a floating point number
float b: a floating point number

Returns:
int: the floor of the sum of two floating point numbers

Constraints
- $0.1 < a, b < 10^6$
- *a and b have at most 8 places after the decimal*

▼ Input Format For Custom Testing

The first line contains a floating point value, *a*
The second line contains a floating point value, *b*

▼ Sample Case 0

Sample Input

```
STDIN     Function
-----     -----
2.3    →  a = 2.3
1.9    →  b = 1.9
```

Sample Output

```
4
```

Explanation
a + b = 2.3 + 1.9 = 4.2. Return the integer floor of the sum, *4*.

▼ Sample Case 1

Sample Input

```
STDIN     Function
-----     -----
```

```
2.34    →  a = 2.34
5.7     →  b = 5.7
```

Sample Output

```
8
```

Explanation

a + b = 2.34 + 5.7 = 8.04. Return the integer floor of the sum, *8*.

Question - 2

Sum Them All

Calculate the sum of an array of integers.

Example

numbers = [3, 13, 4, 11, 9]

The sum is *3 + 13 + 4 + 11 + 9 = 40.*

Function Description

Complete the function *arraySum* in the editor below.

arraySum has the following parameter(s):

int numbers[n]: an array of integers

Returns

int: integer sum of the numbers array

Constraints

- $1 \le n \le 10^4$
- $1 \le numbers[i] \le 10^4$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *numbers*.
Each of the next *n* lines contains an integer *numbers[i]* where $0 \le i < n$.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
5       →   numbers[] size n = 5
1       →   numbers = [1, 2, 3, 4, 5]
2
3
4
5
```

Sample Output 0

```
15
```

Explanation 0

1 + 2 + 3 + 4 + 5 = 15.

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
2       →   numbers[] size n = 2
12      →   numbers = [12, 12]
12
```

Sample Output 1

```
24
```

Explanation 1

12 + 12 = 24.

Question - 3

Prime or Not?

Given an integer, if the number is prime, return 1. Otherwise return its smallest divisor greater than 1.

Example

n = 24

The number *24* is not prime: its divisors are *[1, 2, 3, 4, 6, 8, 12, 24]*. The smallest divisor greater than *1* is *2*.

Function Description

Complete the function *isPrime* in the editor below.

isPrime has the following parameter(s):

long n: a long integer to test

Returns

int: if the number is prime, return *1*; otherwise returns the smallest divisor greater than *1*

Constraints

- $2 \leq n \leq 10^{12}$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The only line of input contains the long integer to analyze, *n*.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
2       →   n = 2
```

Sample Output 0

```
1
```

Explanation 0

As *2* is a prime number, the function returns *1*.

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
4       →   n = 4
```

Sample Output 1

```
2
```

Explanation 1

Since *4* is not a prime number, and the factors of 4 are *[1, 2, 4]*, the function returns the smallest factor of *4* greater than *1*.

Question - 4

Find the Factor

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return *0*.

Example

n = 20

p = 3

The factors of *20* in ascending order are *{1, 2, 4, 5, 10, 20}*. Using 1-based indexing, if *p = 3*, then *4* is returned. If *p > 6*, *0* would be returned.

Function Description

Complete the function *pthFactor* in the editor below.

pthFactor has the following parameter(s):

int n: the integer whose factors are to be found

int p: the index of the factor to be returned

Returns:

int: the long integer value of the p^{th} integer factor of *n* or, if there is no factor at that index, then *0* is returned

Constraints

- $1 \le n \le 10^{15}$
- $1 \le p \le 10^{9}$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number to factor.
The second line contains an integer *p*, the 1-based index of the factor to return.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
10      →   n = 10
3       →   p = 3
```

Sample Output 0

```
5
```

Explanation 0

Factoring $n = 10$ results in $\{1, 2, 5, 10\}$. Return the $p = 3^{rd}$ factor, *5*, as the answer.

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
10      →   n = 10
5       →   p = 5
```

Sample Output 1

```
0
```

Explanation 1

Factoring $n = 10$ results in $\{1, 2, 5, 10\}$. There are only *4* factors and $p = 5$, therefore *0* is returned as the answer.

▼ Sample Case 2

Sample Input 2

```
STDIN       Function
-----       --------
1       →   n = 1
5       →   p = 1
```

Sample Output 2

```
1
```

Explanation 2

Factoring $n = 1$ results in $\{1\}$. The $p = 1^{st}$ factor of *1* is returned as the answer.