Lab session 2: Imperative Programming

For the lab exercises of this week, you are not allowed to use any function from the math library. Nor are you allowed to use arrays (which are dicsussed in week 4)!

The time limit for all problems is set to 1 second, i.e. your program must produce its output within 1 second.

Problem 1: Squareful integers

A positive integer is called a *square free number* if it is not divisible by a square, except (of course) for the square 1. For example, 21 is square free while 75 is not, since it is divisible by $25 = 5^2$. A positive integer is called *squareful* if it is the square of a square free number. For example, $441 = 21^2$ is a squareful number.

Write a program that reads from the input an integer n, where $2 \leq n \leq 2000000000 = 2 \times 10^9$, and outputs whether n is a squareful number. Make sure that your output matches exactly the output as given in the following examples.

Example 1:

input:
```
441
```
output:
```
441 is a squareful number.
```

Example 2:

input:
```
123
```
output:
```
123 is not a squareful number.
```

Example 3:

input:
```
1511887689
```
output:
```
1511887689 is a squareful number.
```

Problem 2: K-composites

We call a positive integer a *k-composite* if it is divisible by the numbers $1, 2, 3, \ldots, k$. For example, the number 2520 is the smallest number that is divisible by $1, 2, 3, .., 9$, and 10.

Write a program that reads from the input a value k (where $1 \leq k \leq 22$), and outputs the smallest k-composite. Note that the smallest 22-composite is 232792560, which fits in a standard `int`. Nevertheless, be careful not to overflow your calculations. It is not allowed to submit a program that contains precomputed answers.

Example 1:

input:
```
3
```
output:
```
6
```

Example 2:

input:
```
5
```
output:
```
60
```

Example 3:

input:
```
10
```
output:
```
2520
```

Problem 3: Counting triples

The input for this problem consists of a single integer n, where $1 \leq n \leq 2000000000 = 2 \times 10^9$. The output of the program should be the number of triples (a, b, c) such that their product equals n, where a, b, and c are positive integers. Note that the order within the triples is irrelevant. For example, the triples $(3, 4, 5)$, $(3, 5, 4)$, $(4, 3, 5)$, $(4, 5, 3)$, $(5, 3, 4)$, and $(5, 4, 3)$ are all considered to be the same triple with the product 60.

For $n = 100$, the output of the program must be 8, since $100 = 1 \cdot 50 \cdot 2 = 25 \cdot 4 \cdot 1 = 1 \cdot 5 \cdot 20 = 1 \cdot 100 \cdot 1 = 10 \cdot 10 \cdot 1 = 25 \cdot 2 \cdot 2 = 2 \cdot 10 \cdot 5 = 4 \cdot 5 \cdot 5$, and there are no other triples.

[Note: Make sure that integer calculations do not overflow.]

Example 1:
input:
```
100
```
output:
```
8
```

Example 2:
input:
```
27
```
output:
```
3
```

Example 3:
input:
```
2000000000
```
output:
```
620
```

Problem 4: Mulpals

A positive integer n is called a *mulpal* (multiplicative palindrome) if there exists an integer d such that $d \cdot n$ equals n read in reverse direction (in decimal notation). For example, $2178 \times 4 = 8712$, so $n = 2178$ is a mulpal.

Write a program that reads two integers a and b from the input (you may assume that $1 \leq a < b < 10000000 = 10^7$), and outputs the number of mulpals n with $a \leq n \leq b$.

Example 1:
input:
```
1 10
```
output:
```
9
```

Example 2:
input:
```
1 100
```
output:
```
18
```

Example 3:
input:
```
1 1000
```
output:
```
108
```