Programming Project 02

This assignment is worth 10 points and must be **completed and turned in before 11:59 on Monday, September 17th , 2017.**

Assignment Overview
This assignment will exercise your ability to utilize control statements for C++ to solve a math problem.

Background
A **perfect** number is an integer whose sum of divisors, excluding the number itself, equals the number. They have been known since at least Euclid in 300 B.C.E, perhaps earlier. The 1st perfect number is 6: its divisors are 1, 2, 3 which, when summed together, equal 6. 28 is the next perfect number with the divisors 1, 2, 4, 7, 14 which, summed together are 28. There are many variations on perfect numbers one of which we will work with for this project, **k-hyperperfect** numbers (see https://en.wikipedia.org/wiki/Hyperperfect_number)

A k-hyperperfect number is a variation on perfect numbers with the following properties:

- We sum all the divisors **excluding 1 and the number**.
- We multiply the number by some factor **k**.
- We add 1 to the product.

If the result is the number back again, the number is k-hyperperfect.

For example, 21 is a 2-hyperperfect number. The divisors are 1,3,7,21 but we ignore the 1 and 21. The sum of the remaining factors (3,7) is 10. Multiply by two and add 1, we get 21. The factors of 301 (ignoring 1 and 301) are 7, 43. ((7+43) * 6) + 1 = 301, making 301 6-hyperperfect. 12211188308281 has the factors 211, 1231, 259741, 47012941, 9919730551, 57872930371. The sum of those is (67839935046 * 180) + 1 equaling 12211188308281 making it 180-hyperperfect.
Look at the table for more examples.

Project Description / Specification
There are two types of test cases in Mimir. Tests where you can see the result (so you can correct your work if you get the incorrect) and those tests where you only get a Pass/Non-Pass answer without seeing the result, so called hidden cases. You will get a combination of both for all Mimir projects from now on. The reason is that we want you to write code that solves the problem according to the specifications, not just give the "correct" answers back as Mimir provides.

Warning: Nonetheless, it is possible to figure out the required answers even on a hidden test case. We require, however, that you write code to solve the problem, not just give back the correct answer. The TAs will check this during grading. The TAs are instructed to give a 0 for any test case that does not solve the problem but only gives back the Mimir required answer.

Input

- Two integers
  - An integer to check if it is k-hyperperfect (if there exists a k that makes the provided integer k-hyperperfect)
  - A max value of k (up to and including the max) that needs to be checked

Output

- If the integer is less than 6 or if the k-max is less than 1, the output is 0
    - Note that no number less than 6 can ever be k-hyperperfect
- If the input integer is k-hyperperfect in the range of 1 to k_max, then the output is that k value. If the input is not k-hyperpefect in the range of 1 to k_max, then the output should be a 0. For example:
    - 21 100 → 2
    - 301 100 → 6
    - 12211188308281 100 → 0
    - 12211188308281 200 → 180
    - -301 100 → 0
    - 21 -5 → 0

Deliverables

proj02/proj02.cpp . The name of the directory is proj02, the name file you turn in should be exactly proj02.cpp. Just like the lab, you must click on proj02, the directory under "Project 02 – k-hyperperfect numbers ", to submit to Mimir. Not the file, the directory.

Notes

- The max value that can be represented by a signed int is ± 2,147,483,647. You need to use a long which has a max of ± 9,223,372,036,854,775,807
- checking all the divisors of some number n by dividing every number less than n into n can be slow. Can you do better. Think about using a square root operation (which is available if you do something like the following)

```
#include<cmath>
…
cout << sqrt(16) << endl;   // result is 4
```