Problem 12. The sequence of triangle numbers is generated by adding the natural numbers. So the 7th triangle number would be $1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$. The first ten terms would be:

$$1, 3, 6, 10, 15, 21, 28, 36, 45, 55, \ldots$$

Let us list the factors of the first seven triangle numbers:

```
1:  1
3:  1,3
6:  1,2,3,6
10: 1,2,5,10
15: 1,3,5,15
21: 1,3,7,21
28: 1,2,4,7,14,28
```

We can see that 28 is the first triangle number to have over five divisors. What is the value of the first triangle number to have over five hundred divisors?

Knowledge Required How to calculate number of divisors of a number in $O(\sqrt{n})$

Solution Outline We start by implementing the `count_divisors` function which takes in a number n and returns the number of divisor in it. This algorithm is a classic one and can be found on the internet. Then we implement the function `triangular_num` which takes in i and returns the i^{th} triangular number.
Then we loop over every i^{th} triangular number starting from 1. We loop until we found out a triangular number whose number of divisors are over 500. We print the number and exit.

Python Solution

```python
def count_divisors(n):
    total = 1
    i = 2
    while i * i <= n:
        if n % i == 0:
            prime_cnt = 0
            while n % i == 0:
                n //= i
                prime_cnt += 1

            total *= (prime_cnt + 1)
        i += 1

    if n > 1:
        total *= 2

    return total

def triangular_num(i):
    return i * (i + 1) // 2

i = 1
while True:
    N = triangular_num(i)
    if count_divisors(N) > 500:
        print(N)
        exit(0)
    i += 1
```