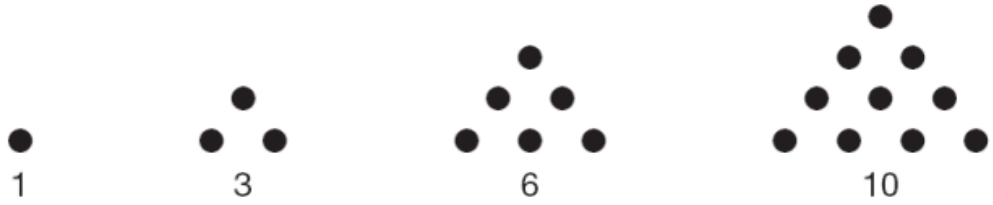
# LAS3006 – Worksheet 2 – Introduction to Streams

**1. Create an infinite `Stream<Integer>` for each of the following mathematical sequence:**

The triangular number series provides the sequence of numbers that can form an equilateral triangle. The n[th] triangle number is the number of balls or dots in a triangle, with n dots on one side.




The formula for this sequence is:

$$T_n = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

So, starting from $n = 0$, the sequence of triangular numbers is:
0, 1, 3, 6, 10, 15, 21, 28, 36, 45

**Hint:** You need to keep track of $n$, so that the next call to your sequence computation function (your lambda expression) can increment it by 1 to get the next value in the sequence. However, the **`Stream<T>.iterate()`** function passes you the last *computed* value, not the last value of $n$. You can create an object that contains the two values ($n$ and $T_n$) and then transform the stream of that object's class to a **`Stream<Integer>`** which provides the sequence of $T_n$.

Print out the first 10 numbers of the sequence through the stream.

**2. Write a simple keyword filter that filters out stop words and duplicates.**
"Stop words" are words which are used in a lot of sentences and thus do not contain significant information with regards to search queries. You are required to filter them out of a stream of words, in order to identify the significant keywords.

a) Create a list of stop words. You can use **`Arrays.asList()`** and pass a list of English stop words such as:
`"a", "an", "the", "so", "it", "do", "did", "for", "by", "of", "to"`

b) Prompt the user to enter a list of words, typing enter after each word. If the user types enter without entering a line it means he is ready from his word list. Put these in an appropriate collection, such as a **`LinkedList<String>`**.

c) Stream the list of words entered by the user and filter out any stop words and duplicates, and print them out sorted in **descending order**.

**Hint:** Have a look at the description of the method **`Comparator.reverseOrder()`**.