Recursion

We're going to practice using recursion using the concept of a triangular number.
A triangular number is the number of objects that can fit in an equilateral triangle with a side made up of *n* objects.

```
- triangle_num 1;
val it = 1 : int
- triangle_num 2;
val it = 3 : int
- triangle_num 3;
val it = 6 : int
```

A triangular number can be calculated using a binomial coefficient: $\binom{n+1}{2}$, or in SML:

```
fun triangle_num(n) =
binom_coeff(n + 1, 2);
```

Recursion: Problem Specification

- Write a function that calculates the nth triangular number.
- You will need to use the following definition of a binomial: $\binom{n}{0} = 1, \binom{0}{k} = 0, \binom{n}{k} = \binom{n-1}{k-1} \times \frac{n}{k}$
- Hint: Turn the above options into function specifications using ML Patterns

`binom_coeff` implementation:

```
fun binom_coeff(n, 0) = 1
| binom_coeff(0, k) = 0
| binom_coeff(n, k) = round(real(
  binom_coeff(n - 1, k - 1)) * (real(n)
  / real(k)));
```

Datatypes

To practice working with both types and datatypes, we're going to do an exercise from the Ullman text, which involves labeled binary trees.

Definition of a labeled binary tree:

```
- datatype 'label btree =
  Empty |
  Node of 'label * 'label btree *
  'label btree;

datatype 'a btree = Empty | Node of
  'a * 'a btree * 'a btree
```

Datatypes: Problem Specification

Define a type (not a datatype) `mapTree` that is a specialization of the `btree` datatype to have a label type that is a set of domain-range pairs.
Type definition:

```
- type ('d, 'r) mapTree = ('d * 'r) btree;

type ('a,'b) mapTree = ('a * 'b) btree
```

Now, define a tree `t1` that has a single node with the pair `("a",1)` at the root.

```
- val t1 = Node(("a",1), Empty, Empty):
  (string, int) mapTree;

val t1 = Node (("a",1),Empty,Empty) :
  (string,int) mapTree
```