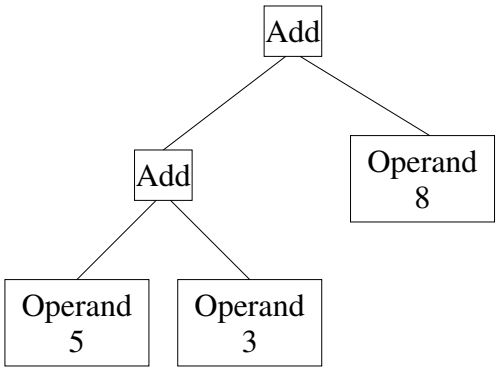
Programming Paradigms Exercise 7 - Haskell 3

Theodoros Chondrogiannis

2nd Semester 2016/17

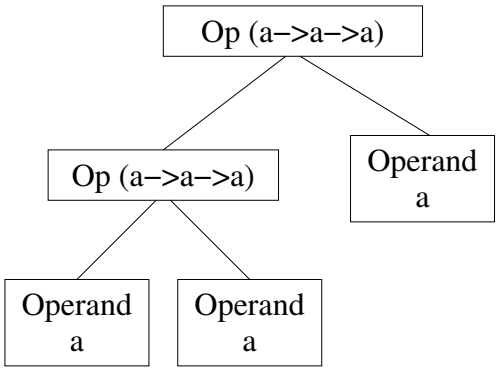
1. Write a function `insertPos` that takes an element `x` and a list `y` as input parameters. It will insert the element `x` into all possible positions of `y`. The result should be a list containing all the lists with `x` inserted into different positions. For example, the input `2` and `[3,5]` should return `[[2,3,5],[3,2,5],[3,5,2]]`.

2. Define a user-defined type for operator trees. An operator tree contains operands of type integer that are connected via the binary operation addition (+). The smallest possible operator tree is one that only contains one operand. The following diagram shows an example:



   Write a function `evaluate` that gets an integer operator tree and evaluates it, i.e., it traverses the tree and adds up all the operands.

3. Now rewrite your user-defined type from part (a) to make it an operator tree that contains operands of any type `a` and any binary operation `a->a->a` defined on type `a`. Again write a function `evaluate` that gets an operator tree and evaluates it.



4. Consider the following two programs you have already implemented.

   - The greatest common divisor program (exercise 5, Q1)
   - The sieve of Eratesthenis (exercise 6, Q2)

   Change your implementations so that you can compile your programs as stand-alone executables. Both programs must get the required arguments from the command line:

```
> ./gcd 12 18
6

> ./sieve 10
[2,3,5,7,9]
```