Часть 1

1. `[1 балл]` Создайте тип для представления логических выражений `BoolExpr`. Ваш тип должен поддерживать логические литералы, отрицание логических выражений, конъюнкцию и дизъюнкцию, равенство **целочисленных выражений**, операцию **больше** для сравнения целочисленных выражений. Целочисленные выражения должны поддерживать: целочисленные литералы, сложение и вычитание. Используя ваш тип, создайте следующие логические выражения:

- `True`
- `not True`
- `True || False`
- `(5+6) > 5`

2. `[1 балл]` Напишите функцию, для вычисления значения логического выражения (True/False)

```
boolEval :: BoolExpr -> Bool
```

Часть 2

Пусть дан тип данных для представления двоичных чисел:

```
data Bin = End | O Bin | I Bin
        deriving (Show, Eq)
```

где
`O` - ноль
`I` - единица

Для удобства будем считать, что двоичные числа записываются, начиная с **младших разрядов** (!!!).
Например:
`O (I (I End))` - представление двоичного числа `110` (`6` в десятичной системе)
`I (I (O End))` - представление двоичного числа `011` (`3` в десятичной системе). Обратите внимание на ведущий ноль. Он ни на что не влияет.

Следующая функция увеличивает двоичное число на единицу.

```
inc :: Bin -> Bin
inc End = I End
inc (O b) = I b
inc (I b) = O (inc b)
```

1. `[0.5 балла]` Напишите реализацию функции, преобразующей двоичное число в `Int`

```
fromBin :: Bin -> Int
fromBin = undefined
```

```
ghci> fromBin (O End)
0
ghci> fromBin (I End)
1
ghci> fromBin (O (I End))
2
ghci> fromBin (I (I End))
3
ghci> fromBin (O (O (I End)))
4
ghci> fromBin (I (O (I End)))
5
ghci> fromBin (I (I (O (O (I (O (I (O End))))))))
83
ghci>fromBin (I (I (O (O (I (O (I End)))))))
83
```

2. `[0.5 балла]` Напишите реализацию функции, преобразующей `Int` в двоичное число

```
toBin :: Int -> Bin
toBin = undefined
```

```
ghci> toBin 0
O End
ghci> toBin 1
I End
ghci> toBin 2
O (I End)
ghci> toBin 3
I (I End)
ghci> toBin 4
O (O (I End))
ghci> toBin 5
I (O (I End))
```

3. `[0.5 балла]` Напишите реализацию функции сложения двух двоичных чисел (при реализации НЕ используйте `toBin` / `fromBin`)

```
pls :: Bin -> Bin -> Bin
pls = undefined
```

```
ghci> pls (O (I End)) (I (O (I End)))
I (I (I End))
```

4. `[0.5 балла]` Напишите реализацию функции умножения двух двоичных чисел (при реализации НЕ используйте `toBin` / `fromBin`)

```
mlt :: Bin -> Bin -> Bin
mlt = undefined
```

```
ghci> mlt (O (I End)) (I (O (I End)))
O (I (O (I End)))
```

5. [1 балл]

Вариант 1 (Если ваш номер в списке группы нечетный!)
Сделайте `Bin` представителем класса типов `Show` так, чтобы строковое представление выглядело как в примере. Не забудьте убрать `deriving Show`. Не печатайте ведущие нули.

```
ghci> End
0
ghci> I End
1
ghci> O (O (I (O (I End))))
10100
ghci> O (O (I (O (I (O End)))))
10100
```

Вариант 2 (Если ваш номер в списке группы четный!)
Сделайте `Bin` представителем класса типов `Eq` так, чтобы сравнение двоичных чисел не учитывало ведущие нули. Например, `I (O (O End))` и `I End` должны считаться равными. Не забудьте убрать `deriving Eq`

```
ghci> O End == End
True
ghci> O (O (I End)) == I End
False
ghci> I (O (O End)) == I End
True
```