Московский физико-технический институт
Факультет инноваций и высоких технологий.
Математическая логика и теория алгоритмов, весна 2013.
Лямбда-исчисление: дополнительные задачи.

Напомним, что мы обозначаем **False** $= \lambda xy.y$ и **True** $= \lambda xy.x$. Нумералами Чёрча для натуральных чисел $0, 1, 2, 3, \ldots$ называются комбинаторы

$$\underline{0} = \lambda fx.x,\ \underline{1} = \lambda fx.fx,\ \underline{2} = \lambda fx.f(fx),\ \underline{3} = \lambda fx.f(f(f(x))), \ldots$$

Напомним также, что комбинатор A представляет функцию $f : \mathbb{N}^k \to \mathbb{N}$, если для любых чисел $n_1, \ldots, n_k$ комбинатор $(A\ \underline{n_1}\ \underline{n_2}\ \ldots\ \underline{n_k})$ равен нумералу Чёрча для числа $f(n_1, \ldots, n_k)$, если $f(n_1, \ldots, n_k)$ определено, и не имеет нормальной формы в противном случае. Аналогично, говорят, что комбинатор B представляет булеву функцию $f : \mathbb{B}^k \to \mathbb{B}$, если каждое произведение B и набора из k комбинаторов **False** и **True** равно соотвествующему данной функции f значению **False** или **True**.

1. Постройте комбинтор **Next** со следующим свойством: для всякого комбинатора F, представляющего некоторую всюду определенную функцию $f\colon \mathbb{N} \to \mathbb{N}$, комбинатор (**Next** F) представляет функцию $g(n) = f(n+1)$.

Указание: На языке `Haskell` нужный нам код мог бы выглядеть совсем просто:

```
let g f = \ y -> f (y+1)
```

2. Постройте комбинатор **Sigma** такой, что для всякого комбинатора F, представляющего некоторую всюду определенную функцию $f\colon \mathbb{N} \to \mathbb{N}$, комбинатор (**Sigma** F) представляет функцию $g(n) = \sum_{i=0}^{n} f(i)$.

Указание: На языке `Erlang` нужный нам код мог бы выглядеть так:

```
sigma(F) ->
  fun(X) ->
    if
      X==0 -> F(0) ;
      true -> F(X)+ (sigma(F))(X-1)
    end
  end.
```

3. Постройте комбинатор **3Negation** со следующим свойством: для всякого комбинатора F, представляющего некоторую булеву функцию трёх аргументов $f(x_1, x_2, x_2)$, комбинатор (**3Negation** F) представляет отрицание функции f.

4. Постройте комбинатор **Inverse** такой, что для всякого комбинатора F, представляющего некоторую всюду определенную функцию $f\colon \mathbb{N} \to \mathbb{N}$, комбинатор (**Inverse** F) представляет функцию $g(n) = \min\{x \ : \ f(x) = n\}$. *Замечание:* функция g может быть не всюду определена.

Указание: На языке `Scala` нужный нам код могла бы выглядеть так:

```
def minGEq(f:Int=>Boolean,n:Int):Int = {
  if(f(n)) n else minGEq(f,n+1)
}
def inverse(f:Int=>Int):Int=>Int = {
  x:Int => minGEq(y=>f(y)==x,0)
}
```

5. Постройте комбинатор **Prime** такой, что

$$\textbf{Prime}\ \underline{n} = \begin{cases} \textbf{True}, & \text{если число } n \text{ простое,} \\ \textbf{False}, & \text{иначе} \end{cases}$$

(проверка числа на простоту).

6. Постройте комбинатор **NthPrime**, который по нумералу Чёрча n находит n-ое простое число. (Для определённости потребуем, чтобы **NthPrime** $\underline{0} = \underline{1}$.)

7. Постройте комбинатор **GCD**, представляющий в λ-исчислении функцию вычисления наибольшего общего делителя двух натуральных чисел.

Указание. Программа для вычисления наибольшего общего делителя на языке `OCaml` может выглядеть так:

```
let rec gcd n m =
  if      n = 0 then m
  else if m = 0 then n
  else if n > m then gcd m (n - m)
  else               gcd n (m - n)
```

8. Функция Аккермана $\mathsf{A}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ определяется как

$$\mathsf{A}(m,n) = \begin{cases} n+1, & \text{если } m = 0, \\ \mathsf{A}(m-1,1), & \text{если } m > 0 \text{ и } n = 0, \\ \mathsf{A}(m-1,\mathsf{A}(m,n-1)), & \text{если } m > 0 \text{ и } n > 0. \end{cases}$$

Вычислите $\mathsf{A}(0,0)$, $\mathsf{A}(1,1)$, $\mathsf{A}(2,2)$, $\mathsf{A}(3,3)$. Постройте комбинатор **A**, представляющий функцию Аккремана в λ-исчислении. *Указание:* Программа на языке `Scheme` для вычисления функции Аккермана может выглядеть так:

```
(define (A m n)
    (if (= m 0)
        (+ n 1)
        (if (= n 0)
            (A (- m 1) 1)
            (A (- m 1) (A m (- n 1))))))
```

9. В *стрелочной нотации Кнута* для положительных целых чисел a, b используются обозначения $a \uparrow b = a^b$ (a в степени b), $a \uparrow\uparrow b = \underbrace{a \uparrow (a \uparrow (\ldots \uparrow a)\ldots)}_{b \text{ раз}}$,

$$a \underbrace{\uparrow\uparrow \ldots \uparrow}_{n} b = \underbrace{a \underbrace{\uparrow\uparrow \ldots \uparrow}_{n-1} (a \underbrace{\uparrow\uparrow \ldots \uparrow}_{n-1} (\ldots \underbrace{\uparrow\uparrow \ldots \uparrow}_{n-1} a)\ldots)}_{a \text{ встречается } b \text{ раз}};$$

n-ым числом Аккермана называют $Ack_n = n \underbrace{\uparrow\uparrow \ldots \uparrow}_{n} n$. Вычислите первые три числа Аккермана. Постройте комбинатор **Ack**, который преобразует нумерал $\underline{n}$ (для $n > 0$) в нумерал, выражающий n-ое число Аккермана.