Programação Funcional, 2024.2 — **Prova 1**

(Turma T12 do Thanos) — (points: 100; bonus: $0^\flat$; time: 84')

Nome:

2024-10-30

Regras:

I. Não vires esta página antes do começo da prova.

II. Nenhuma consulta de qualquer forma.

III. Nenhum aparelho ligado (por exemplo: celular, tablet, notebook, *etc.*).[1]

IV. Nenhuma comunicação de qualquer forma e para qualquer motivo.

V. $\forall x\big(\text{Colar}(x) \to \neg\text{Passar}(x, \text{FUN})\big)$.

VI. Escreva em forma facilmente legível.

VII. Responda dentro das caixas indicadas.

VIII. Entregue *todas* as folhas de rascunho extra, juntas com tua prova.

IX. Nenhuma prova será aceita depois do fim do tempo!

Cada uma dessas vale **4** pts no **B**:

```
odd  :: Nat → Bool          min   :: Nat → Nat → Nat
(+)  :: Nat → Nat → Nat     prod  :: List Nat → Nat
(*)  :: Nat → Nat → Nat     len   :: List a → Nat
```

Cada uma dessas vale **6** pts no **B**:

```
take   :: Nat → List a → List a         dropWhile :: (a → Bool) → List a → List a
tails  :: List a → List (List a)        concat    :: List (List a) → List a
repeat :: a → List a                    (++)      :: List a → List a → List a
any    :: (a → Bool) → List a → Bool    replicate :: Nat → a → List a
```

Cada uma dessas vale **8** pts no **B**:

```
zip       :: List a → List b → List (a,b)            pairs     :: List a → List (a,a)
subseqs   :: List a → List (List a)                  inits     :: List a → List (List a)
zipWith   :: (a → b → c) → List a → List b → List c  countDown :: Nat → List Nat
```

Boas provas!

[1] Ou seja, *desligue antes* da prova.

(35) # A

Defina os: `Nat`, `List`, `map`, `filter`, `fold`, `curry`, `uncurry`.
DEFINIÇÕES.

(32) # B

Escolha **até 4** das funções da primeira página par definir. É **proibido** usar list comprehension. Veja *bem* os tipos, pois podem ser diferentes dos escolhidos pelo Prelude da Haskell.
DEFINIÇÕES.

(12) # C

Defina `zip` em termos da `zipWith` e `zipWith` em termos da `zip`.

(21) # D

Considere o tipo `Int` dado, junto com suas operações.

(7) **D1.** Defina um tipo de dados `ArEx` para representar expressões de aritmética de inteiros formadas apenas pelas operações binárias $(+)$ e $(\cdot)$ e a operação unária $(-\_)$.

(7) **D2.** Defina uma função `eval` de *evaluate* que dada uma expressão aritmética de inteiros retorna seu valor-resultado.

(7) **D3.** Defina uma função `height` que retorna a altura da árvore sintáctica da sua entrada.

Só isso mesmo.

Rascunho