Paradigmas de Linguagens Computacionais
Prova 1 -- 20/05/2015 – página 1

Aluno:_____

1) (2 pontos) Apresente a inferência dos tipos das seguintes expressões (função f e função g), e dê um exemplo de uso das mesmas com parâmetro(s)/argumento(s) e um resultado.

```
f ::
f = map (+)

g ::
g = (.) map map
```

para ajudar a resolver a questão, abaixo estão os tipos de map, (.) e (+):

```
map :: (t -> u) -> [t] -> [u]
(.) :: (b -> c) -> (a -> b) -> (a -> c)
(+) :: Int -> Int -> Int
```

2) Um sorteio da Mega-Sena pode ser representado por uma lista de seis números. Um conjunto de cartões de apostas pode ser representado por uma lista de listas (cada lista representando um cartão).
Assuma que os número do resultado premiado/sorteado e os números em cada cartão estão ordenados.
Defina funções para:

(1.0 ponto) a função premiados retorna o numero de cartões premiados com a sena.

```
type Resultado = [Int]
type Jogos = [[Int]]

premiados :: Resultado -> Jogos -> Int
```

(1.0 ponto) a função acertos retorna a lista com o número de acertos em cada cartão (do primeiro cartão, do segundo, do terceiro, etc.).

```
acertos :: Resultado -> Jogos -> [Int]
```

(1.0 ponto) a função numPremios retorna uma tupla de três inteiros contendo a quantidade de cartões premiados com 4, 5 ou 6 acertos, respectivamente.

```
numPremios :: Resultado -> Jogos -> (Int, Int, Int)
```

Aluno:_____

3) Uma linguagem de programação baseada em pilha possui apenas uma pilha (stack) onde ficam os dados/operandos e todas as instruções são apenas de empilhar, desempilhar ou fazer operações consumindo (lendo) os dados no topo da pilha e deixando o resultado final o topo da pilha.
Dados os tipos de dados abaixo e os exemplos, escreva um interpretador que executa as instruções com o comportamento abaixo:

```
data Instrucao  = PUSH Int -- empilha um valor inteiro
                | POP      -- desempilha (remove) um valor do topo da pilha
                | ADD      -- remove (lê) os dois valores no topo da pilha e
deixa a soma deles no topo da pilha
                | SUB      -- remove (lê) os dois valores no topo da pilha e
deixa a soma deles no topo da pilha|
                | DUP      -- repete o mesmo valor no topo da pilha (duplica
ele)

type Pilha = [Int]
```

(1.0 ponto) evalI avalia uma única instrução em uma dada pilha

```
evalI :: Instrucao -> Pilha -> Pilha
-- exemplos: eval ADD [1,2,3,4,5] ---> [3,3,4,5] (soma 1+2)
-- exemplos: eval DUP [5,1] ---> [5,5,1] (repete/copia o valor no topo da pilha)
-- exemplos: eval SUB [1,2,3,4] ---> [-1,3,4] -- calcula 1-2=-1
-- exemplos: eval ADD [1,2,3,4,5] ---> [3,3,4,5] (soma 1+2)
-- exemplos: eval PUSH 7 [1,2,3] ---> [7,1,2,3] insere o 7 no topo
-- exemplos: eval POP [8,2,3] ---> [2,3] -- remove 8
```

(1.0 ponto) evalProg avalia um programa (sequência de instruções) a partir de uma pilha inicial vazia e retorna o estado final da pilha depois da avaliação

```
evalProg :: [Instrucao] -> Pilha
-- exemplos: evalProg [PUSH 3, PUSH 5, DUP, ADD, SUB] ---> [7] (5+5-3)
```

4) (2.0 pontos) faça uma função translate que traduz expressões (tipo Expr, abaixo) para uma sequência (lista) de instruções que, se usadas com o avaliador da questão 4, avaliam a expressão.
Exemplo:

```
data Expr = Literal Int -- um número
          | Soma Expr Expr -- soma as duas expressões
          | Subtrai Expr Expr -- subtrai a segunda expressão da primeira
          | Dobra Expr -- dobra o valor da expressão

translate :: Expr -> [Instrucao]
translate (Soma (Literal 5)
                (Dobra (Subtrai (Literal 4) (Literal 1))))
----> [PUSH 1, PUSH 4, SUB, DUP, ADD, PUSH 5, ADD]
```

5) (1.0 ponto) qual a diferença entre avaliação estrita e preguiçosa (lazy)? Mostre exemplos desta diferença.