Teste Final

Programação Funcional – 1º Ano, MIEI / LCC / MIEF

11 de Janeiro de 2016 **Duração: 2 horas**

1. Apresente uma definição recursiva das seguintes funções sobre listas:
   (a) `nub :: Eq a => [a] -> [a]` que calcula uma lista com os mesmos elementos da recebida, sem repetições. Por exemplo, `nub [1,2,1,2,3,1,2]` corresponde a `[1,2,3]`.
   (b) `zipWith :: (a->b->c) -> [a] -> [b] -> [c]` que combina os elementos de duas listas usando uma função específica. Por exemplo, `zipWith (+) [1,2,3,4,5] [10,20,30,40]` corresponde a `[11,22,33,44]`.

2. Considere o tipo `MSet a` para representar multi-conjuntos de elementos de `a`

   ```
   type MSet a = [(a,Int)]
   ```

   Considere ainda que nestas listas não há pares cuja primeira componente coincida, nem cuja segunda componente seja menor ou igual a zero.
   (a) Defina a função `converte :: Eq a => [a] -> MSet a` que converte uma lista para um multi-conjunto. Por exemplo, `converte "bacaba"` corresponde a `[('b',2),('a',3),('c',1)]`
   (b) Defina a função `intersect :: Eq a => MSet a -> MSet a ->MSet a` que calcula a intersecção de dois multi-conjuntos. Por exemplo,
   `intersect [('a',3),('b',5),('c',1)] [('d',5),('b',2)]` corresponde a `[('b',2)]`

3. Considere o seguinte tipo para representar expressões proposicionais:

   ```
   data Prop = Var String | Not Prop | And Prop Prop | Or Prop Prop

   p1 :: Prop
   p1 = Not (Or (And (Not (Var "A")) (Var "B")) (Var "C"))
   ```

   (a) Declare `Prop` como instância da classe `Show`, de forma a que a expressão `p1` seja apresentada da seguinte forma: `-((-A /\ B) \/ C)`
   (b) Defina a função `eval :: [(String,Bool)] -> Prop -> Bool`, que dada uma *valoração* (i.e., uma correspondência entre variáveis proposicionais e valores booleanos) calcula o valor lógico de uma expressão proposicional.
   (c) Uma proposição diz-se na *forma normal negativa* se as negações só estão aplicadas às variáves proposicionais. Por exemplo, a proposição $((A \vee \neg B) \wedge \neg C)$ está na forma normal negativa, enquanto `p1` não está.
   Defina a função `nnf :: Prop -> Prop` que recebe uma proposição e produz uma outra que lhe é equivalente, mas que está na forma normal negativa. Por exemplo, o resultado de `nnf p1` deverá ser a proposição $((A \vee \neg B) \wedge \neg C)$.
   Lembre-se das seguintes leis: $\neg\neg A = A$, $\neg(A \vee B) = \neg A \wedge \neg B$ e $\neg(A \wedge B) = \neg A \vee \neg B$.
   (d) Defina a função `avalia :: Prop -> IO Bool` que, dada uma proposição, pergunta ao utilizador a valoração de cada variável presente na proposição e calcula com essa informação o seu valor lógico.