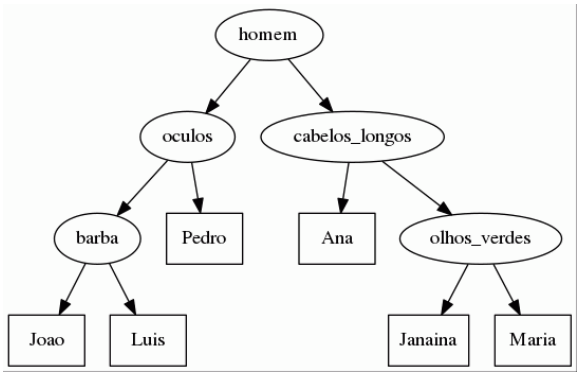
Obs.: Os codigos das respostas podem ser acessados em: https://repl.it/@luis_poli/LPF-2Av-2018-1

2ª Avalliação de Linguagem de Programação Funcional

Nome: _____

Uma árvore de decisão é uma técnica de programação utilizada para classificar objetos.
Os nós desta árvore podem significar indecisão ou decisão. Os nós indecisos possuem como informação uma propriedade do objeto e dois ramos (indicado as decisões se o objeto possui ou não a propriedade). Os nós de decisão não possuem filhos e contem a classificação do objeto. Por exemplo, na árvore:



Vemos que se uma pessoa é Homem, Usa óculos e Barba, então deve ser o João.

1) Escreva um programa para implementar uma árvore de decisões e uma função de classificação, que recebe uma lista de propriedades (lista de strings) do objeto e retorna a sua classificação (ou a String “” se nenhuma classificação for determinada).

```
data DecisionTree =
   Question String DecisionTree DecisionTree  | Class String

busca :: DecisionTree -> [String] -> String
busca (Question q sim nao) props =
   if (elem q props) then
      busca sim props  else busca nao props
busca (Class c) props = c
```

2) A implementação default da classe Show para esse tipo de dados imprimiriam os dados da árvore de forma difícil de entender. Uma implementação mais desejável poderia produzir uma lista de classes e suas propriedades. Por exemplo:

Joao => barba, oculos, homem
Ana => cabelos_longos
etc…

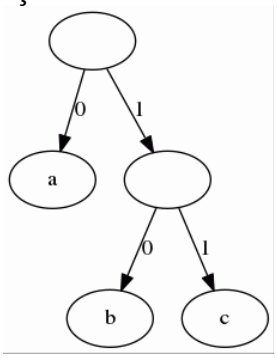
Escreva uma implementação da classe Show para seu tipo de dados que produza a saída desejada.

```
instance Show DecisionTree where
  show d = aux "" d where
    aux prefix (Question q sim nao) =  (aux (q ++ " " ++ prefix) sim) ++ "\n" ++  (aux prefix nao)
    aux prefix (Class c) = c ++ " : " ++ prefix
```

2ª Avalliação de Linguagem de Programação Funcional

Nome: _____

Uma árvore de Huffman é uma técnica de programação utilizada para decodificar objetos compactados usando uma codificação de tamanho variável. Por exemplo, na árvore abaixo:



o caractere “a” é codificado como a sequencia de bits “0”, enquanto o caractere “c” é codificado como a sequencia “11”.

1) Escreva um programa para implementar uma árvore de huffman e uma função de decodificação, que recebe uma sequencia de bits (string com caracteres 0 e 1) que codifica um ou mais caracteres e retorna a sequencia de caracteres decodificada.

```
data HuffTree = Node HuffTree HuffTree | Code Char

decode :: HuffTree -> String -> String
decode raiz str = aux raiz str where
  aux (Code c) str = c : (aux raiz str)
  aux tree "" = ""
  aux (Node zero um) ('0':str) = aux zero str
  aux (Node zero um) ('1':str) = aux um str
```

2) A implementação default da classe Show para esse tipo de dados imprimiriam os dados da árvore de forma difícil de entender. Uma implementação mais desejável poderia produzir uma lista de classes e suas propriedades. Por exemplo:

```
a => 0
b => 1 0
c => 1 1
```

Escreva uma implementação da classe Show para seu tipo de dados que produza a saída desejada.

```
instance Show HuffTree where
  show h = aux "" h where
    aux prefix (Code c) = c : (" = " ++ prefix)
    aux prefix (Node zero um) =
      (aux (prefix ++ " 0") zero) ++ "\n" ++
      (aux (prefix ++ " 1") um)
```

Boa Sorte,
Profs. Luis Menezes e Thiago Farias