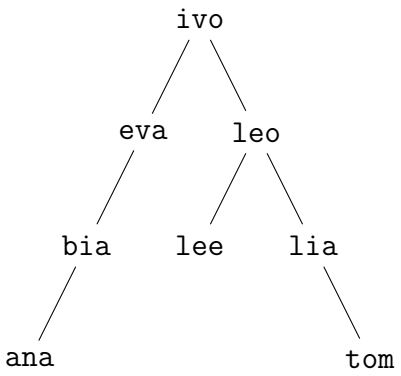
Estruturas de Dados
Prática: listas encadeadas

1. (4 pontos) Sabemos que um percurso *Em-Ordem* em uma árvore binária de busca produz uma listagem dos seus elementos em ordem crescente. Como podemos modificar esse percurso para obeter uma ordenação decrescente? Apresente uma implementaçõa e explique brevemente o seu funcionamento.

2. (6 pontos) Percorrer uma árvore binária *em ordem de nível* significa começar na raiz e, em seguida, visitar todos os nós com distância 1 da raiz (começando com o nó esquerdo, se não for nulo); e, em seguida, visitar todos os nós com distância 2 da raiz (novamente, da esquerda para a mais direita); e depois visitar todos os nós com distância 3 e assim por diante.

   Escreva um método para percorrer uma árvore binária por níveis. Observação você vai precisar de uma estrutura de dados auxiliar para armazenar os nós que estão em um mesmo nível à medida que você se afasta da raiz.

   Tomemos por exemplo a árvore abaixo. A saída esperada é: ivo, eva, leo, bia, lee, lia, ana e tom.



Código de apoio

```
class ArvBinariaBusca:

  #construtor
  def __init__(self, dado=None):
    self.esq = None
    self.dir = None
    self.dado = dado

  # transforma  rvore  em string (somente para impressao)
  def __repr__(arv):
    return ("[" + str(arv.dado) + ",  " + str(arv.esq) + ", " + str(arv.
                                            dir) +
            "]")

  # insere um novo item na  rvore
  def inserir(raiz, dado):
    if not raiz.dado:
      raiz.dado = dado
    else:
      # nao permite elementos repetidos
      if raiz.dado == dado:
        return
      # para que lado ir ?
      elif raiz.dado < dado:
        # se existe sub-arvore a direita
        if raiz.dir:
          raiz.dir.inserir(dado)
        else:
          raiz.dir = ArvBinariaBusca(dado)
      else:
        # se existe sub-arvore a esq
        if raiz.esq:
          raiz.esq.inserir(dado)
        else:
          raiz.esq = ArvBinariaBusca(dado)
    return

  # Percurso Em Ordem
  def em_ordem(raiz):
    # visita sub-arvore esquerda
    if raiz.esq:
      raiz.esq.em_ordem()
    # visita raiz
    print(raiz.dado)
    # visita sub-arvore direita
    if raiz.dir:
      raiz.dir.em_ordem()
```