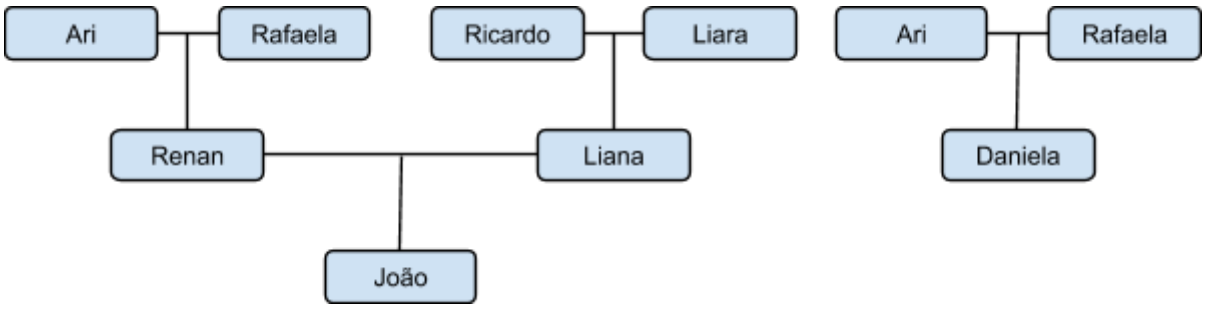
MC202 — ESTRUTURAS DE DADOS

Lab 07 — Árvores Binárias

Problema

A consanguinidade é definida como a afinidade entre pessoas que apresentam ascendência comum. Em outras palavras, são consideradas consanguíneas pessoas que tenham ao menos um ancestral em comum. Pode-se medir o quanto uma pessoa é consanguínea com outra por meio do chamado **grau de consanguinidade**. Neste laboratório, iremos considerar o método de contagem germânico que, para contar o grau consanguinidade entre duas pessoas, conta-se o número de níveis até algum antepassado em comum na árvore genealógica ascendente (que tem ascendências paterna e materna) de uma pessoa. Por exemplo, primos diretos possuem grau 2 de consanguinidade através de um avô em comum. Note que pode haver uma diferença geracional entre as pessoas consideradas. Observe o exemplo a seguir, que contém as árvores ascendentes de João e Daniela.



No exemplo acima, João é consanguíneo com Daniela em grau 2 por meio de Ari e também por meio de Rafaela. No outro sentido, Daniela é consanguínea com João em grau 1 por meio de Ari e Rafaela.

Entrada

São dadas como entrada duas **árvores binárias com todos os níveis completos** que apresentam a genealogia ascendente de duas pessoas. A primeira linha contém dois inteiros positivos H_1 e H_2 que indicam, respectivamente, a altura de cada uma das árvores. As duas linhas seguintes correspondem ao resultado do percurso **pós-ordem** de cada uma das árvores. Cada nó é uma sequência de caracteres com no mínimo 1 e no máximo 20 caracteres. Em uma árvore não há nós repetidos, i.e., todas as sequências de caracteres presentes em uma árvore são distintas.

Saída

O seu programa deve produzir duas linhas de saída. Cada linha corresponde ao resultado do percurso **pré-ordem** em cada uma das árvores da entrada, mas apenas os nós que representam algum ancestral em comum são impressos, para as duas pessoas para as quais as árvores genealógica foram dadas. Junto ao nó, deve ser impresso também o grau de consanguinidade e um espaço após este número.

Exemplo 1

Entrada

```
3 2
Ari Rafaela Renan Ricardo Liara Liana Joao
Ari Rafaela Daniela
```

Saída

```
Ari 2 Rafaela 2
Ari 1 Rafaela 1
```

Exemplo 2

Entrada

```
3 3
Mateus Rafaela Marcos Ricardo Liara Liana Joao
Ricardo Julia Ana Antonio Fatima Mateus Aline
```

Saída

```
Mateus 2 Ricardo 2
Ricardo 2 Mateus 1
```

Observações

- Neste laboratório, é obrigatório implementar e utilizar árvores binárias de acordo com a struct que utiliza dois ponteiros para os filhos de um nó vista em aula.
- Para as turmas A, B e C, esse laboratório tem peso 3.
- Deverão ser submetidos os seguintes arquivos: **`lab07.c`**, **`arv_binaria.h`** e **`arv_binaria.c`**. Outros 2 (dois) arquivos **opcionais e de sua escolha** podem ser

submetidos, desde que contribuam para a solução e que mantenham ou melhorem a qualidade do código.

- Observações sobre SuSy:
  - Versão do GCC: 4.4.7 20120313 (Red Hat 4.4.7-17).
  - Flags de compilação:
    - `-ansi -Wall -pedantic-errors -Werror -g -lm`
  - Utilize comentários do tipo `/* comentário */`;
    comentários do tipo `//` serão tratados como erros pelo SuSy
  - Tempo máximo de execução: 1 segundo.
  - Evite usar o sistema como compilador ou apenas para testar as instâncias.
  - Sempre use o software valgrind para detectar erros decorrentes do uso incorreto da memória dinâmica **em todos os testes abertos**. Relembrando:
    - `valgrind --leak-check=full ./lab07 < arq01.in`
- Além das observações acima, esse laboratório será avaliado pelos critérios:
  - Indentação de código e outras boas práticas, tais como:
    - uso de comentários (apenas quando são relevantes);
    - código simples e fácil de entender;
    - sem duplicidade (partes que fazem a mesma coisa).
  - Organização do código:
    - arquivo de cabeçalho (.h) contendo definições de constantes, tipos de dados criados pelo usuário, protótipo de funções etc;
    - arquivos de implementação (.c) que implementam as funções definidas no arquivo de cabeçalho;
    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
  - Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - desalocar toda memória alocada dinamicamente durante a execução do programa;
    - se não realiza leitura ou escrita em blocos de memória não alocados;
    - inicialização de variáveis sempre que for necessário;
    - dentre outros critérios.