# Primeiro Trabalho Prático

## CI1057 - 2023/2

**André Guedes**

### Implementação da Árvore AVL

O objetivo do trabalho é a implementação das rotinas de manipulação de uma *árvore AVL*. O trabalho deve ter uma biblioteca e uma aplicação.

## Biblioteca:

A biblioteca do trabalho deve ter as definições de uma árvore AVL que armazene números inteiros e as funções de busca, inserção e remoção, e possivelmente outras.

O nome da biblioteca deve ser `avl-tree`.

## Aplicação:

Uma aplicação para usar a biblioteca deve ser feita. Esta aplicação será um pequeno programa que recebe instruções textuais e as executa, gerando uma resposta. Neste programa a árvore inicialmente está vazia, a cada comando a árvore é modificada, e ao final do arquivo de entrada o programa termina.

## Entrada:

A entrada deve conter comandos. Os comandos são `i` (inserção), `b`, (busca) e `r`, (remoção). Todos os comando tem um parâmetro, um número inteiro, que aparece separado do comando por um espaço. Os comandos aparecem um por linha.

Exemplo de entrada:

```
i 1
i 2
i 3
i 4
b 5
r 4
```

## Saída:

A cada comando executado uma saída deve ser gerada. A saída deve representar o estado da árvore após a execução do comando. Para o comando de busca, a saída deve também indicar se o número procurado está ou não na árvore, e apresentar a lista de nós consultados. O formato da descrição da árvore é "(R,E,D)", onde R é o valor guardado no nó raíz, E e D são as sub-árvores esquerda e direita (recursivamente). A árvore vazia é representada por "()". Um exemplo pode ser visto abaixo.

Para representar uma árvore onde a raíz é o "a", "a" tem fihos "b" e "c". "b" não tem filhos e "c" só tem o filho esquerdo "d", usamos a seguinte linha:

```
(a,(b,(),()),(c,(d,(),()),()))
```

Perceba que não existem espaços entre os símbolos.

A saída, para cada comando de alteração ("i" ou "r"), será a repetição do comando em uma linha (tal como foi lido da entrada) e na linha seguinte a descrição da árvore resultante.

A saída para o comando de busca será a repetição do comando em uma linha e na linha seguinte a seqüência de nós visitados, separados por vírgulas (",") e mais nada. Caso o nó procurado tenha sido encontrado ele será o último da seqüência.

Exemplo de saída (para a entrada acima):

```
i 1
(1,(),())
i 2
(1,(),(2,(),()))
i 3
(2,(1,(),()),(3,(),()))
i 4
(2,(1,(),()),(3,(),(4,(),())))
b 5
2,3,4
r 4
(2,(1,(),()),(3,(),()))
```

# Requisitos mínimos:

O trabalho deve ser feito de forma que possa ser compilado e executado nas servidoras de computação do Departamento de Informática.

O nome do executável deve ser `busca`.

Não deve ter nenhuma opção de linha comando. A entrada deve ser lida da entrada padrão (\textsc{stdin}) e o resultado deve ser escrito na saída padrão (\textsc{stdout}).

# O que deve ser entregue:

Além dos arquivos fonte, deve acompanhar um `makefile` e um arquivo `README` explicando o que foi feito e com o nome dos autores. Qualquer particularidade deve estar descrita neste texto.

Para compilar vou usar o comando `make` (sem nenhum parâmetro), portanto preparem o Makefile para fazer isso.

Para testar vou rodar um script como o abaixo.

```
busca < teste.in > teste.out
diff teste.sol teste.out
```

Caso o teste seja positivo (não imprime nada) será analisado o código fonte.

# Forma de entrega:

O trabalho deve ser empacotado em um arquivo login1-login2.tar.gz, onde login1-login2 é uma string com os logins dos integrantes da equipe nas servidoras do DInf. Ao descompactar este arquivo o `make` e o script acima deverão funcionar dentro do diretório local (não em subdiretórios).

Este arquivo deve ser enviado por e-mail ao endereço do professor (andre) com o assunto "CI1057-trab1" (exatamente).

# Equipe:

O trabalho pode ser feito em equipes de até 2 (dois) alunos.