Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Informática (DAINF)

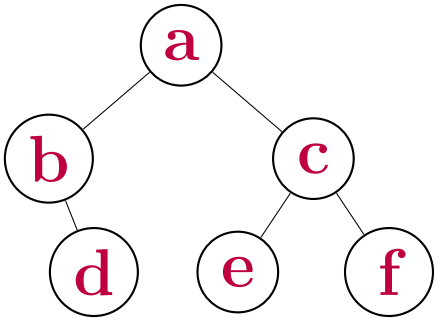
Estruturas de dados II

Professora: Juliana de Santi (jsanti@utfpr.edu.br)

Lista de exercícios

Exercício 1) Considerando a árvore a seguir, defina a saída para os percursos:

- pré-ordem
- in-ordem
- pós-ordem



Exercício 2) Considerando a árvore do exercício 1, escreva funções para os percursos:

- pré-ordem
- in-ordem
- pós-ordem

```
Arvore *a = constroi_arv (‘a’,
  constroi_arv(‘b’,
    cria_arv_vazia(),
    constroi_arv(‘d’,cria_arv_vazia(),cria_arv_vazia())
  ),
  constroi_arv(‘c’,
    constroi_arv(‘e’,cria_arv_vazia(),cria_arv_vazia()),
    constroi_arv(‘f’,cria_arv_vazia(),cria_arv_vazia())
  )
);
```

Exercício 3) Escreva uma função que retorna um valor booleano (um ou zero) que indica a ocorrência ou não de um dado caractere na árvore. Considere o seguinte protótipo para a sua função:

```
int pertence_arv (Arvore *a, char c);
```

onde char **c** é o caractere que deve ser procurado na árvore **a**.

Exercício 4) Escreva uma função que conte o número de nós de uma árvore binária. Utilize o seguinte protótipo para a sua função:

```
int conta_nos (Arvore *a);
```

Exercício 5) Escreva uma função que calcula a altura de uma árvore binária. Utilize o seguinte protótipo para a sua função:

```
int calcula_altura_arvore (Arvore *a);
```

Exercício 6) Escreva uma função que conta o número de nós folhas em uma árvore binária. Utilize o seguinte protótipo para a sua função:

```
int conta_nos_folha (Arvore *a);
```

Os exercícios a seguir (*) são exercícios extra, ou seja, não fazem parte da lista para pontuação. Não haverá lista-solução e o estudante que quiser resolve-los, deverá pensar a solução por conta própria.

Exercício *) Para descrever árvores binárias, podemos usar a seguinte notação textual: a árvore vazia é representada por <>, e árvores não-vazias, por < raiz esq dir >. Com essa notação, a árvore do Exercício 1 é representada por:

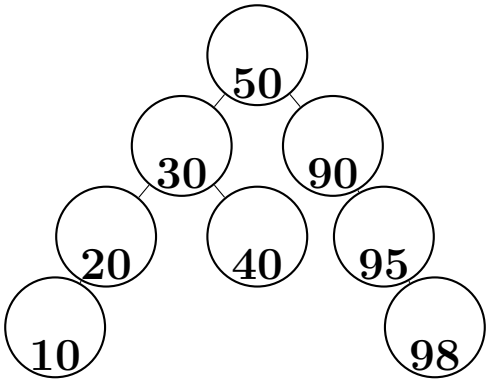
é representada por:

```
<a<b<><d<><><>><><c<e<><><>><f<><><>>>>
```

Escreva uma função que recebe uma árvore de entrada e a imprime utilizando essa notação. Protótipo:

```
void imprime_arv_marcadores (Arvore* arv)
```

Exercício *) Suponha a árvore binária abaixo. (a) ache o grau de cada um dos nós. (b) determine os nós folhas. (c) complete a árvore com quantos nós forem necessários, para transformá-la em uma árvore binária cheia.



Exercício *) Escreva uma função que imprime e retorna o maior valor inteiro em uma árvore binária composta por valores inteiros. Utilize o seguinte protótipo para a sua função:

```
int max_arvore (Arvore *a);
```

Exercício *) Escreva uma função que recebe uma árvore binária como entrada e imprime seus nós em níveis. Por exemplo, para a árvore do Exercício 1, a impressão desejada é representada por:

```
a b c d e f
```

Assinatura/protótipo da função:

```
void imprime_por_nivel(Arvore *a)
```

Exercício *) Escreva uma função que verifica se uma dada árvore binária é cheia (ou completa). Utilize o seguinte protótipo para a sua função (retorna 1 se cheia, 0 caso contrário):

```
int verifica_arvore_cheia (Arvore *a);
```

Exercício *) Duas árvores são espelho-similares se elas são vazias ou se elas não são vazias e suas subárvores esquerda de cada uma são espelho-similares as subárvores direita da outra (Fig. 1).

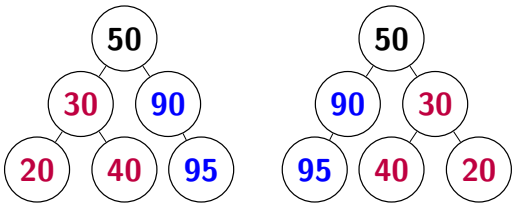


Figura 1: Árvore binária espelho-similar.

Escreva uma função que faz o espelho de uma árvore binária. Utilize o seguinte protótipo para a sua função:

```
Arvore espelha_arvore (Arvore *a);
```