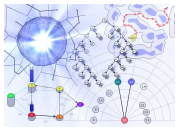
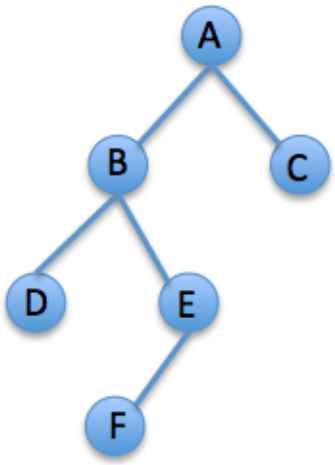


Frequência 2

Atenção: Para todos os exercícios do teste podem ser implementadas funções auxiliares, desde que transcritas para a folha de prova.

1. (3 valores) Considere a seguinte árvore binária. Esquematize a árvore que resulta de 2 operações de rotação à direita.



2. Considere a seguinte estrutura de dados binária:

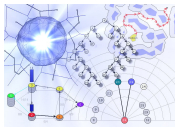
```
typedef struct NODO{
        int id;                 %critério de ordenação na árvore
        char nome[80];
        struct NODO *fe;
        struct NODO *fd;
}Nodo;
```

   a. (3 valores) Implemente uma função que verifique se existem activos (diferentes de NULL) mais filhos esquerdos ou direitos na árvore (-1=esquerdos, 0=igual, 1=direitos).

   Protótipo: int maisFilhosActivos(Nodo* A);
   //A=raiz

   b. (3 valores) Implemente uma função que mostre o nome dos elementos com ID num determinado intervalo.

   Protótipo: void mostraNomeIntervalo(Nodo* A, int idInf, int idSup);
   //A=raiz. idInf, idSup= limites inferior e superior do intervalo



c. (3 valores) Codifique uma função que verifique se uma árvore está contida noutra, isto é, se os ID de todos os elementos da primeira árvore também estão na segunda árvore.

Protótipo: int arvoreContida(Nodo* A1, Nodo* A2);
//A1=raiz da sub-árvore, A2=raiz da árvore

3. (4 valores) Grafos: Considere um grafo representado através de listas ligadas de adjacência. Implemente uma função que receba informação de 1 caminho (sequência de vértices) e devolva o total de vértices do grafo pelos quais o caminho não passa.

Protótipo: int verticesNaoVisitados(Nodo v, int totV, int *cam, int totC);**
//v=Grafo, totV=total de Vértices, cam=sequência de vértices, totC=total de vértices no caminho.

4. (4 valores) Dijkstra. Considere a seguinte estrutura de dados, relativa à execução do algoritmo de Dijkstra. Implemente uma função que receba uma estrutura "Dijkstra" já preenchida e verifique se um determinado vértice pertence ao caminho mais curto entre o vértice origem e destino.

```
typedef struct{
        float *custo;
        int *visitado;
        int *predecessor;
}Dijkstra;
```

Prótotipo: int pertenceCaminhoMaisCurto(Nodo **v, int totV, Dijsktra D, int vert, int vo, int vd);
//v=Grafo, totV=total de vértices, D=caminho mais curto entre "vo" e "vd", vert=vértice a verificar, vo=vértice origem, vd=vértice destino
// (1=Sim, 0=Não)

Exemplo:
custo=[0 10 15 12 25 14 27],
visitado=[1 1 1 1 1 1 1],
predecessor=[-1 0 1 1 2 1 4]

vo=0, vd=6, vert=4 pertence ao caminho mais curto vo→vd, vert=5 não pertence ao caminho mais curto vo→vd.