UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
Departamento de Ciências de Computação

SCC0122 Estruturas de Dados
Prof. Thiago A. S. Pardo

Prova 2
11/12/2020

Responda as questões da prova no editor de texto de sua preferência e submeta a resolução no e-Disciplinas (aba "Avaliações"). Se desejar, você pode responder nesse próprio documento. Opcionalmente, você também pode responder em papel e submeter um arquivo com a digitalização/foto da prova.

Nome do aluno: Gustavo Siqueira Barbosa

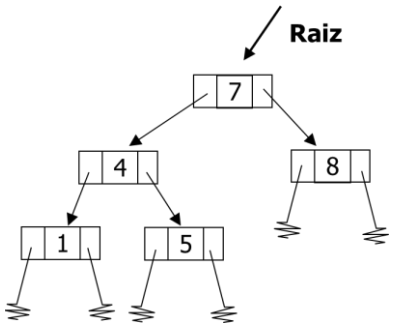
Número USP: 10728122

ATENÇÃO! Parte das respostas se encontram após a última página de questões

1) Considere as árvores binárias de busca (não balanceadas) e a declaração de sua estrutura abaixo.

```
typedef int elem;

typedef struct bloco {
        elem info;
        struct bloco *esq, *dir;
} no;

typedef struct {
        no *raiz;
} ABB;
```

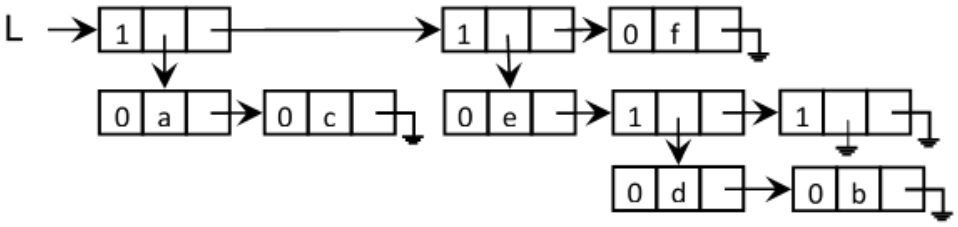
(a) (2,0) Implemente em C uma função que retorne a soma dos elementos armazenados em uma árvore passada por parâmetro. Se a árvore estiver vazia, a função deve retornar zero como resposta. Por exemplo, se receber a árvore abaixo, a função deve retornar o valor 25. Faça as demais suposições que julgar necessárias.



(b) (1,0) Considerando a árvore de exemplo do item anterior, indique a ordem de impressão dos elementos da árvore segundo cada um dos percursos abaixo. Um item já foi feito para você.

Percurso	**Elementos impressos**
Em-ordem	1, 4, 5, 7, 8
Pré-ordem	7, 4, 1, 5, 8
Pós-ordem	1, 5, 4, 8, 7
Largura	7, 4, 8, 1, 5
Profundidade	7, 4, 1, 5, 8

2) (2,0) Uma lista generalizada L é uma sequência finita de n≥0 nós, sendo que cada nó armazena um átomo ou uma sub-lista, o que é indicado pelo campo "tipo" do nó: se tipo=0, então o nó armazena um átomo; se tipo=1, então o nó armazena uma sub-lista. A figura abaixo é um exemplo de representação da lista generalizada [[a,c],[e,[d,b],[]],f].



Considere a declaração abaixo da estrutura de dados dessa lista.

```
typedef struct bloco {
        union {
                char atomo;
                struct bloco *sublista;
        } info;
        int tipo;
        struct bloco *prox;
} no;

typedef struct {
        no *inicio;
} ListaGen;
```

Implemente em C uma função que retorne o número de nós com átomos em uma lista generalizada. No exemplo anterior, a função deveria retornar 6.

3) Considerando os grafos eulerianos, responda:

(a) (0,5) Quais as condições para um grafo ser considerado euleriano?

(b) (0,5) Desenhe um grafo euleriano.

(c) (0,5) Desenhe um grafo que tenha um caminho euleriano, mas que não é um grafo euleriano.

(d) (0,5) Quais as condições para um grafo ser considerado hamiltoniano?

4) (3,0) Suponha que exista uma rede social específica para cada turma de graduação da USP. Considere que essa rede seja representada em um grafo, onde os vértices numerados representam os alunos da turma e as arestas indicam os amigos verdadeiros dentro da turma. Por exemplo, se João (cujo número hipotético na rede é 1) e Maria (cujo número hipotético é 7) forem amigos verdadeiros, então deve haver uma aresta entre os vértices correspondentes no grafo. Suponha também que esse grafo (não direcionado e não ponderado) é representado como uma matriz de adjacências, conforme declaração abaixo:

```
#define NumAlunosNaTurma 50

typedef struct {
        int m[NumAlunosNaTurma][NumAlunosNaTurma];
} Grafo;
```

No exemplo dos amigos verdadeiros João e Maria, as posições m[1][7] e m[7][1] deveriam ser iguais a 1, indicando que há relação de amizade verdadeira entre eles.

Implemente uma função em C que receba o número de um aluno e retorne o número de seu amigo verdadeiro que é mais popular, ou seja, que tenha mais amigos verdadeiros na rede. Se houver empate, retorne o primeiro que encontrar. Faça as demais suposições que julgar necessárias.

1)

a)

```
int EstaVazia(ABB *A){
    if (A->raiz == NULL) return 1;
    else return 0;
}

int SomaElementos(ABB *A){
    if (EstaVazia(A)) return 0;
    return (aux_SomaElementos(A->raiz));
}

int aux_SomaElementos(no *node){
    int soma = node->info;

    if (node->esq != NULL) soma += aux_SomaElementos(node->esq);

    if (node->dir != NULL) soma += aux_SomaElementos(node->dir);

    return soma;
}
```

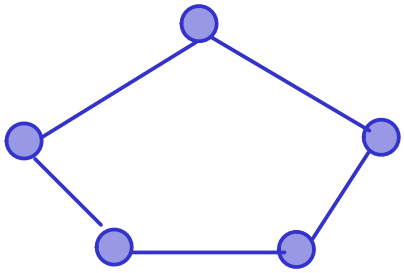
2)

```
int EstaVazia(ListaGen *L){
    if (L->inicio == NULL) return 1;
    else return 0;
}

int numAtomos(ListaGen *L){
    if (EstaVazia(L)) return 0;
    return (aux_numAtomos(L->inicio));
}

int aux_numAtomos(no *node){
    if (node == NULL) return 0;

    int n_atomos = 0;

    if (node->tipo == 1) n_atomos += aux_numAtomos(node->info.sublista);
    else n_atomos++;

    if (node->prox != NULL) n_atomos += aux_numAtomos(node->prox);

    return n_atomos;
}
```

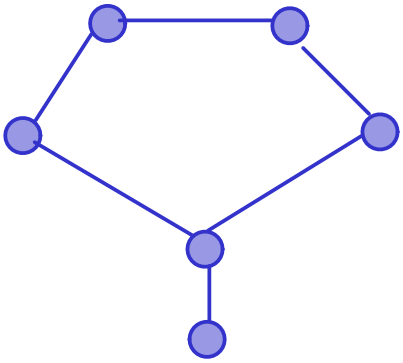
3)

a) Um grafo euleriano é um grafo onde se é possível percorrer um ciclo euleriano. Em outras palavras, trata-se de um grafo onde se pode percorrer um caminho passando por todas as arestas de forma que se inicie e encerre no mesmo vértice.

b)



c)



d) Para que um grafo seja considerado um grafo hamiltoniano, deve ser possível percorrer-se um ciclo hamiltoniano. Em outras palavaras, deve ser possível passar por todos os vértices do grafo apenas uma vez, de forma que o caminho se inicie e se encerre em um mesmo vértice.

4)

```
/*Assumo que o numero do aluno passado representa seu índice. Portanto, um número
  que seja 0 é válido e todos os menores que 0 e maiores ou igual a 50, inválidos. A função
   retorna -1 quando o aluno não tiver nenhum amigo, e -2 quando a entrada for inválida
*/
int NumAlunoPopular(Grafo *G, int n_aluno){

    if (n_aluno >= NumAlunosNaTurma || n_aluno < 0){
        printf("ERRO! Número inválido!\n");
        return -2;
    }

    int n_aluno_popular = -1;   //armazena o índice do aluno mais popular;
    int soma_aluno_popular = 0; //armazena a quantidade de amigos do aluno mais popular;
    int soma;   //soma auxiliar para comparacao

    for (int i=0; i < NumAlunosNaTurma; i++){

        if (G->m[n_aluno][i] == 1){
            soma = 0;

            for (int j=0; j < NumAlunosNaTurma; j++) if (G->m[i][j] == 1) soma++;

            if (soma > soma_aluno_popular){
                soma_aluno_popular = soma;
                n_aluno_popular = i;
            }
        }
    }

    return n_aluno_popular;
}
```