

Fundação CECIERJ - Vice Presidência de Educação Superior a Distância  
**Curso de Tecnologia em Sistemas de Computação**  
**Disciplina: EAD05032 - Programação Orientada a Objetos**  
**AP3 – 1º semestre de 2023.**


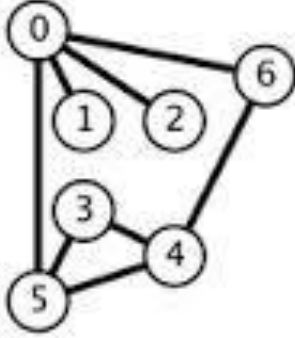
Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

**Questão 1) (5,0 PONTOS)**

Escreva um método em JAVA que receba um grafo (que não deve ser alterado pela sua função) e retorne **true** se do nó **origem** é possível chegar ao nó **destino** com, no máximo, **k** arestas, onde **k**  $\geq 1$ , e **false** caso contrário.

Exemplos de entradas e saídas são mostradas a seguir:

GRAFO EM ARQUIVO E DEMAIS ENTRADAS	REPRESENTAÇÃO GRÁFICA*	SAÍDA
1 2 1 2 2 1 Se origem = 1, destino = 2 e k = 2		<b>true</b>
0 1 2 3 4 5 6 0 1 1 0 0 2 2 0 0 5 5 0 0 6 6 0 3 4 4 3 3 5 5 3 4 5 5 4 4 6 6 4		<b>false</b>

Se origem = 1, destino = 4 e k = 2	
------------------------------------	--

\*Desenhos obtidos da Internet.

Use as seguintes classes no desenvolvimento de sua resposta:

```
import java.io.*;

class Vizinho{
    int no_viz;
    Vizinho prox;

    Vizinho(int c){
        no_viz = c;
        prox = null;
    }

    public String toString(){ return no_viz + " "; }
}

class Lista{
    int no;
    Vizinho prim_viz;
    Lista prox_no;

    Lista(int c){
        no = c;
        prim_viz = null;
        prox_no = null;
    }

    Vizinho pertence(int no){
        Vizinho resp = prim_viz;
        while((resp != null) && (no != resp.no_viz))
            resp = resp.prox;
        return resp;
    }

    void ins_Viz(int c){
        Vizinho v = pertence(c);
        if(v != null) return;
        v = new Vizinho(c);
        v.prox = prim_viz;
        prim_viz = v;
    }

    public String toString(){
        String resp = no + ":\n";
        Vizinho p = prim_viz;
        while(p != null){
            resp += p.toString();
            p = p.prox;
        }
        return resp + "\n";
    }
}

class Grafo{
    Lista prim;

    Grafo(){ prim = null; }
```

```

Lista pertence(int no){
    Lista resp = prim;
    while((resp != null) && (no != resp.no)) resp = resp.prox_no;
    return resp;
}

void insere(int no1, int no2){
    Lista p = pertence(no1);
    p.ins_Viz(no2);
}

void insere(int no){
    Lista p = pertence(no);
    if(p == null){
        p = new Lista(no);
        Lista q = prim;
        if(q == null){
            prim = p;
            return;
        }
        while(q.prox_no != null) q = q.prox_no;
        q.prox_no = p;
    }
}

public String toString(){
    String resp = "";
    Lista p = prim;
    while(p != null){
        resp += p.toString();
        p = p.prox_no;
    }
    return resp;
}
}

class ListaNo{
    Vizinho prim;

    ListaNo(){ prim = null; }

    void insere(int x){
        Vizinho novo = new Vizinho(x);
        novo.prox = prim;
        prim = novo;
    }

    boolean busca(int x){
        Vizinho p = prim;
        while(p != null){
            if(p.no_viz == x) return true;
            p = p.prox;
        }
        return false;
    }
}

```

```

    }

    public String toString(){
        String resp = "";
        Vizinho p = prim;
        while(p != null){
            resp += p.toString();
            p = p.prox;
        }
        return resp + "\n";
    }
}

public class Q1_AP3_POO_2023_1{
    public static void main(String[] args) throws IOException{
        BufferedReader in;
        in = new BufferedReader(new FileReader(args[0]));
        int origem = Integer.parseInt(args[1]);
        int dest = Integer.parseInt(args[2]);
        int k = Integer.parseInt(args[3]);
        try {
            Grafo g = new Grafo();
            String s, vs[];
            s = in.readLine();
            vs = s.split(" ");
            for(int i = 0; i < vs.length; i++)
                g.insere(Integer.parseInt(vs[i]));
            while((s = in.readLine()) != null){
                vs = s.split(" ");
                g.insere(Integer.parseInt(vs[0]), Integer.parseInt(vs[1]));
            }
            in.close();

            //IMPLEMENTAR ESSE METODO E RETIRAR COMENTARIO PARA TESTAR
            //System.out.println(teste(g, origem, dest, k));
        }
        catch (Exception e){
            System.out.println("Excecao\n");
        }
    }
}

```

#### RESPOSTA:

```

public class Q1_AP3_POO_2023_1{
    public static void main(String[] args) throws IOException{
        //IDEM ENUNCIADO DA PROVA: RETIRAR COMENTÁRIO DA LINHA
        //System.out.println(teste(g, origem, dest, k));
    }
    public static boolean teste(Grafo g, int origem, int destino, int k){
        if(k <= 0) return false;
        Lista p = g.pertence(origem);
        if(p == null) return false;
        if((k >= 1) && (p.pertence(destino) != null)) return true;
        if(k == 1) return false;
        Vizinho v = p.prim_viz;
        while(v != null){
            boolean resp = teste(g, v.no_viz, destino, k - 1);
            if(resp) return resp;
            v = v.prox;
        }
    }
}

```

```
    }  
    return false;  
}  
}
```

### Questão 2) (5,0 PONTOS)

Suponha que você esteja ajudando o CEDERJ no desenvolvimento de uma aplicação que tenha por objetivo gerar um relatório de disciplinas cursadas pelos estudantes. Neste relatório, para cada disciplina existente, deve-se informar a maior média entre os estudantes que cursaram a disciplina e o nome deste estudante. As disciplinas serão apresentadas no relatório de saída em ordem decrescente de média. O dado de entrada é um arquivo texto que registra cada disciplina cursada por estudante, com o seu nome e sua respectiva média obtida. Um exemplo de um arquivo de entrada é mostrado a seguir:

```
INF1001/Fulano das Couves/7.3  
INF1620/Sicrano da Silva/6.7  
INF1620/Beltrano Raimundo/8.4  
INF1001/Sicrano da Silva/8.7  
INF1620/Fulano das Couves/7.2
```

Escreva um programa que leia o arquivo de entrada, gere um arquivo de saída, cujo nome é “saida-maior-” acrescido do nome do arquivo de entrada, com as informações agrupadas por disciplina. Para o exemplo de entrada fornecido acima, o arquivo de saída seria:

```
INF1001 Maior nota: 8.7 Nome: Sicrano da Silva  
INF1620 Maior nota: 8.4 Nome: Beltrano Raimundo
```

Um exemplo de uso desse programa seria `java calcMediaDisc notas.txt`, onde `notas.txt` é o nome do arquivo de entrada e `saida-maior-notas.txt` o nome do arquivo de saída.

## Gabarito:

```
import java.io.*;
import java.util.*;

public class Disciplina implements Comparable<Disciplina> {
    public Disciplina(String codigoDisciplina, String nomeAluno, float cr) {
        this.codigoDisciplina = codigoDisciplina;
        this.nomeAluno = nomeAluno;
        this.cr = cr;
    }
    public String getCodigoDisciplina() {
        return codigoDisciplina;
    }
    public String getNomeAluno() {
        return nomeAluno;
    }
    public void setNomeAluno(String nomeAluno) { this.nomeAluno = nomeAluno; }
    public float getCR() { return cr; }
    public void setCR(float cr) { this.cr = cr; }
    @Override
    public int compareTo(Disciplina outra) {
        if (this.cr < outra.cr) return 1;
        else if (this.cr > outra.cr) return -1;
        else return 0;
    }
    private String codigoDisciplina;
    private String nomeAluno;
    private float cr;
}

public class CalcMediaDisc {
    public static void main(String[] args) throws IOException {
        // Criar uma lista vazia de disciplinas.
        ArrayList<Disciplina> disciplinas = new ArrayList<>();
        // Preencher a lista de disciplinas usando os dados de entrada. Ou seja...
        BufferedReader in = new BufferedReader(new FileReader(args[0]));
        try {
```

```
// ... para cada linha do arquivo de entrada...
String linha;
while ((linha = in.readLine()) != null) {
    // ... quebrar a linha em três valores ...
    String valores[] = linha.split("/");
    String codigoDisciplina = valores[0];
    String nomeAluno = valores[1];
    float cr = Float.parseFloat(valores[2]);
    // ... e buscar a disciplina na lista de disciplinas.
    Disciplina disciplina = null;
    for (int i = 0; i != disciplinas.size(); ++i) {
        if (disciplinas.get(i).getCodigoDisciplina().equals(codigoDisciplina)) {
            disciplina = disciplinas.get(i);
            break;
        }
    }
    // Caso a disciplina exista, então atualizar seus dados, se necessário.
    if (disciplina != null) {
        if (disciplina.getCR() < cr) {
            disciplina.setNomeAluno(nomeAluno);
            disciplina.setCR(cr);
        }
    }
    // Caso contrário, criar a disciplina com o aluno atual.
    else {
        disciplinas.add(new Disciplina(codigoDisciplina, nomeAluno, cr));
    }
}
}
finally {
    in.close();
}
// Agora basta colocar a lista de disciplinas em ordem decrescente.
// O enunciado não proíbe o uso de implementações prontas de ordenação.
Collections.sort(disciplinas);
// Por fim, escrever o arquivo de saída.
BufferedWriter out = new BufferedWriter(new FileWriter("saida-maior-" + args[0]));
try {
```

```
        for (int i = 0; i < disciplinas.size(); ++i) {
            Disciplina disciplina = disciplinas.get(i);
            out.write(disciplina.getCodigoDisciplina() + " Maior nota: " + disciplina.getCR() + " Nome: "
+ disciplina.getNomeAluno() + "\n");
        }
    }
    finally {
        out.close();
    }
}
```