Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

**Curso de Tecnologia em Sistemas de Computação**
**Disciplina: Programação Orientada a Objetos**
**AP1 2° semestre de 2018.**

**Nome –**

**Assinatura –**

---

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) (3.0 pontos)**

Defina uma classe de objetos que representam uma data (dia, mês e ano), contendo os seguintes métodos:

a) um método toString, que retorna uma cadeia de caracteres correspondente à data representada pelo objeto alvo da chamada, no formato "dd/mm/aaaa", onde dd, mm e aaaa correspondem, respectivamente, ao dia, mês e ano desta data;

b) um método compara, que compara a data representada pelo objeto alvo da chamada com uma data passada como argumento para o método; o valor retornado deve ser 0 se essas datas são iguais, ou um número negativo se a primeira data é anterior à última, ou um número positivo se a primeira é posterior à última.

RESPOSTA:

```
class Data {
        int dia;
        int mes;
        int ano;

        public Data(int dia, int mes, int ano) {
                this.dia = dia;
                this.mes = mes;
                this.ano = ano;
        }

        public String toString() {
```

```
                return dia + "/" + mes + "/" + ano;
        }

        public int compara (Data d) {
                if (dia == d.dia && mes == d.mes && ano == d.ano)
                        return 0;
                if (ano < d.ano ||
                                (ano == d.ano && mes < d.mes) ||
                                        (ano == d.ano && mes == d.mes && dia < d.dia))
                        return -1;
                return 1;
        }
}
```

**Questão 2) (3.0 pontos)**

a) Crie uma classe para representar reuniões. Objetos desta classe devem possuir dia, mês e ano da reunião, assim como o assunto da reunião, hora de início e de fim. Para simplificar, pode-se assumir horas como valores inteiros, onde início é sempre menor que fim.

b) Além de definir seus campos, defina um construtor para a classe de forma adequada. Defina também um método chamado *getDuracao()*, o qual obtém o tempo de duração de uma reunião.

c) Numa classe main(), crie as seguintes reuniões listadas na tabela abaixo, adicione num vetor, percorra-o e exiba a quantidade total de horas das reuniões listadas.

| Assunto | Hora Início | Hora Fim | Dia | Mês | Ano |
|---|---|---|---|---|---|
| Reunião com Sócios | 15 | 17 | 13 | 8 | 2018 |
| Reunião com  Clientes | 17 | 18 | 13 | 8 | 2018 |
| Confraternização | 21 | 23 | 13 | 8 | 2018 |

RESPOSTA:

```
class Reuniao extends Data {
        String assunto;
        int horaInicio;
        int horaFim;

        public Reuniao(String a, int hi, int hf, int dia, int mes, int ano) {
                super(dia, mes, ano);
                this.assunto = a;
                this.horaInicio = hi;
                this.horaFim = hf;
        }

        public int getDuracao() {
                return this.horaFim - this.horaInicio;
        }
}

public class AP1_2018_2_Q2 {
        public static void main(String[] args) {
                Reuniao r1 = new Reuniao("Reunião com sócios", 15, 17, 13, 8, 2018);
```

```
                Reuniao r2 = new Reuniao("Reunião com clientes", 17, 18, 13, 8, 2018);
                Reuniao r3 = new Reuniao("Confraternização", 21, 23, 13, 8, 2018);
                Reuniao reunioes[] = {r1, r2, r3};
                int duracao = 0;
                for (Reuniao r : reunioes) {
                        duracao += r.getDuracao();
                }
                System.out.println("O tempo total de reunião é: " + duracao);
        }
}
```

**Questão 3) (4.0 pontos)**

Seu programa em Java deverá fazer a seguinte ordenação: serão dados **N** números e um inteiro positivo **M (N e M serão informados usando a Classe Scanner)**. Este programa terá que ordenar estes **N** números em ordem ascendente de seu módulo **M**. Se houver um empate entre um número ímpar e um número par (para os quais o seu módulo **M** fornece o mesmo valor), então o número impar precederá o número par. Se houver um empate entre dois números ímpares (para os quais o seu módulo **M** fornece o mesmo valor), então o maior número ímpar precederá o menor número ímpar. Se houve um empate entre dois números pares (para os quais o seu módulo **M** fornece o mesmo valor), então o menor número par precederá o maior número par.

A entrada contém vários casos de teste. Cada caso de teste inicia com dois inteiros. **N** ($0 < \mathbf{N} \leq 10000$) e **M** ($0 < \mathbf{M} \leq 10000$) que denotam quantos números existirão neste conjunto. Depois serão recebidos **N** números. A entrada é terminada por uma linha que conterá dois zeros e não deve ser processada. **Valores negativos para N e M serão desconsiderados.**

A saída ocorrerá da seguinte maneira: a primeira linha de cada conjunto de saída conterá os valores de **N** e **M**. As próximas **N** linhas conterão os **N** números, ordenados de acordo com as regras acima mencionadas. Imprima os dois últimos zeros da entrada para a saída padrão.

| **Exemplos de Entrada** | **Exemplos de Saída** |
|---|---|
| 15 3<br>1<br>2<br>3<br>4<br>5<br>6<br>7<br>8<br>9<br>10<br>11<br>12<br>13<br>14<br>15 | 15 3<br>15<br>9<br>3<br>6<br>12<br>13<br>7<br>1<br>4<br>10<br>11<br>5<br>2<br>8<br>14 |

| | |
|---|---|
| 3 3<br>9<br>12<br>10 | 3 3<br>9<br>12<br>10 |
| 0 0 | 0 0 |

**RESPOSTA:**

```
import java.util.Scanner;

class Elem{
  int num, resto;

  Elem(int n, int r){
    num = n;
    resto = r;
  }

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

public class Q3_AP1_2018_2{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt(), m = sc.nextInt();
    while((n != 0) && (m != 0)){
      int i, aux;
      Elem vet[] = new Elem[n];
      for(i = 0; i < n; i++){
        aux = sc.nextInt();
        vet[i] = new Elem(aux, aux % m);
      }
      Ordena(vet);
      System.out.println(n + " " + m);
      for(i = 0; i < n; i++) System.out.println(vet[i]);
      n = sc.nextInt();
      m = sc.nextInt();
    }
    System.out.println("0 0");
  }

  /*
   -> Se houver um empate entre um número ímpar e um número par (para os quais o
      seu módulo M dá o mesmo valor), então o número impar irá preceder o número
      par.
   -> Se houver um empate entre dois números ímpares (para os quais o seu módulo
     M dá o mesmo valor), então o maior número ímpar irá preceder o menor ímpar.
   -> Se houve um empate entre dois números pares (para os quais o seu módulo M
     dá o mesmo valor), então o menor número par irá preceder o maior par.
  */
  public static void Ordena(Elem vet[]){
    int i;
    for(i = 0; i < vet.length; i++){
      int j, prox = i;
      for(j = i + 1; j < vet.length; j++){
        if(vet[j].resto < vet[prox].resto)
          prox = j;
        else{
           if((vet[j].resto == vet[prox].resto) && ((vet[j].num + vet[prox].num)
% 2 == 1) && (vet[j].num % 2 == 1))
            prox = j;
```

```
          else{
             if((vet[j].resto == vet[prox].resto) && (vet[prox].num % 2 == 1) &&
(vet[j].num % 2 == 1) && (vet[j].num > vet[prox].num))
               prox = j;
             else{
                if((vet[j].resto == vet[prox].resto) && (vet[prox].num % 2 == 0)
&& (vet[j].num % 2 == 0) && (vet[j].num < vet[prox].num))
                 prox = j;
             }
           }
         }
       }
       if(prox != i){
         Elem temp = vet[i];
         vet[i] = vet[prox];
         vet[prox] = temp;
       }
     }
   }
}
```