Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II
Profs.: Felipe Domingos da Cunha, Max do Val Machado e Rodrigo Richard Gomes

Trabalho Prático I

1 Regras Básicas

1. Estude bastante cada par de entrada/saída.
2. Todos os programas serão desenvolvidos nas linguagens Java ou C. Quando não especificada a linguagem, considere que a mesma é Java.
3. Todas as apresentações de trabalho devem ser feitas no Linux usando o editor VIM.
4. Como o combinado não sai caro: As cópias de trabalho serão encaminhadas para o colegiado; O aluno perderá 1 ponto para cada método não comentado ou com comentário inútil.
5. As exceções devem ser perguntadas/discutidas/negociadas com o professor.
6. Na primeira parte deste trabalho, nas questões em Java, você não pode usar os métodos das classes String, Integer, Double, ... Os únicos métodos permitidos são *char charAt(int)* e *int length()* da classe String.
7. Para cada questão, devemos submeter apenas um arquivo (.java ou .c). Essa regra será necessária para a submissão de trabalhos no Verde e no identificador de plágio utilizado na disciplina.
8. Use o padrão Java para comentários.
9. Na leitura de um número inteiro, se seu valor for vazio ou não número, ele recebe zero.
10. Nos exercícios de ordenação ou estruturas de dados, se dois registros tiverem a mesma chave de pesquisa, eles serão ordenados pelo atributo nome.
11. O arquivo **readme.txt** contém dicas sobre Linux, VIM e redirecionamento de entrada/saída.
12. Para contar as letras, consoantes e vogais desconsidere acentos e cedilha, ou seja, consideramos somente os caracteres cujos códigos ASCII estiverem entre 'A' e 'Z' ou 'a' e 'z'.
13. No Java, não utilize as classes IO nem Scanner.

14. Para cada exercício: faça (entende-se análise, implementação e comentários), teste (várias vezes) e submeta no Verde. Os exercícios não testados/comentados serão penalizados.

15. A correção será automática através do Verde e de entrevista com o aluno nas aulas de laboratório.

16. A correção sempre será feita na versão do Linux disponível nos laboratórios. Qualquer problema devido ao uso de outras arquiteturas será de responsabilidade do aluno.

17. Fique atento ao Charset dos arquivos de entrada e saída.

2 Dicas sobre Recursividade

1. Um método recursivo pode ser usado como uma estrutura de repetição e um erro básico consiste em confundir tais estruturas com os métodos recursivos. Para evitar esse erro, neste trabalho, não podemos usar estruturas de repetição nos métodos recursivos.

2. Outro erro básico consiste em criar variáveis globais (em Java, atributos) para darem suporte à recursividade. Para evitar esse erro, neste trabalho, não podemos usar variáveis globais.

3. Um terceiro erro básico consiste em passar o valor de retorno como parâmetro. Para evitar esse erro, neste trabalho, não podemos passar parâmetros extras aos do enunciado, exceto um contador para recursividade. O método abaixo contém um exemplo desse erro.

```
boolean contarLetrasMinusculas(String s, int i, int resp){
   int resp;
   if(i == s.length()){
      resp = 0;
   }
   else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false){
      resp = contarLetrasMinusculas(s, i + 1, resp + 1);
   }
   else {
      resp = contarLetrasMinusculas(s, i + 1, resp);
   }
   return resp;
}
```

4. Quando criamos um método recursivo, normalmente, passamos um contador como parâmetro. Uma forma elegante de implementar essa solução recursiva consiste em criar dois métodos: o recursivo; e outro que chama o recursivo pela primeira vez e inicializa o valor do contador. Neste trabalho, devemos usar essa técnica em todos os métodos recursivos. Veja o exemplo abaixo:

```

boolean somenteLetrasMinusculas(String s){
   return somenteLetrasMinusculas(s, 0);
}
boolean somenteLetrasMinusculas(String s, int i){
   boolean resp;
   if(i == s.length()){
      resp = true;
   }
   else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false){
      resp = false;
   }
   else {
      resp = somenteLetrasMinusculas(s, i + 1);
   }
   return resp;
}
```

5. Uma boa prática de programação é colocar um único *return* em cada método. Por exemplo, o método anterior é mais indicado que a versão abaixo.

```
1
2  boolean somenteLetrasMinusculas(String s, int i){
3      if(i == s.length()){
4          return true;
5      }
6      else if ((s.charAt(i) >= 'a' && s.charAt(i) <= 'z') == false){
7          return false;
8      }
9      else {
10         return somenteLetrasMinusculas(s, i + 1);
11     }
12 }
```

3 Questões

1. **Palíndromo** - Crie um método **iterativo** que recebe uma string como parâmetro e retorna true se essa é um palíndromo. Na saída padrão, para cada linha de entrada, escreva uma linha de saída com SIM / NÃO indicando se a linha é um palíndromo. Destaca-se que uma linha de entrada pode ter caracteres não letras.

2. **Palíndromo em C** - Refaça a questão anterior na linguagem C.

3. **Ciframento de César** - O Imperador Júlio César foi um dos principais nomes do Império Romano. Entre suas contribuições, temos um algoritmo de criptografia chamado "Ciframento de César". Segundo os historiadores, César utilizava esse algoritmo para criptografar as mensagens que enviava aos seus generais durante as batalhas. A ideia básica é um simples deslocamento de caracteres. Assim, por exemplo, se a chave utilizada para criptografar as mensagens for 3, todas as ocorrências do caractere 'a' são substituídas pelo caractere 'd', as do 'b' por 'e', e assim sucessivamente. Crie um método **iterativo** que recebe uma string como parâmetro e retorna outra contendo a entrada de forma cifrada. Neste exercício, suponha a chave de ciframento três. Na saída padrão, para cada linha de entrada, escreva uma linha com a mensagem criptografada.

4. **Alteração Aleatória** - Crie um método **iterativo** que recebe uma string, sorteia duas letras minúsculas aleatórias (código ASCII $\geq$ 'a' e $\leq$ 'z'), substitui todas as ocorrências da primeira letra na string pela segunda e retorna a string com as alterações efetuadas. Na saída padrão, para cada linha de entrada, execute o método desenvolvido nesta questão e mostre a string retornada como uma linha de saída. Abaixo, observamos um exemplo de entrada supondo que para a primeira linha as letras sorteados foram o 'a' e o 'q'. Para a segunda linha, foram o 'e' e o 'k'.

EXEMPLO DE ENTRADA:	EXEMPLO DE SAÍDA:
o rato roeu a roupa do rei de roma	*o rqto roeu q roupq do rei de romq*
e qwe qwe qwe ewq ewq ewq	*k qwk qwk qwk kwq kwq kwq*
FIM	

   A classe Random do Java gera números (ou letras) aleatórios e o exemplo abaixo mostra uma letra minúscula na tela. Em especial, destacamos que: i) *seed* é a semente para geração de números aleatórios; ii) nesta questão, por causa da correção automática, a *seed* será quatro; iii) a disciplina de Estatística e Probabilidade faz uma discussão sobre "aleatório".

```
1 Random gerador = new Random();
2 gerador.setSeed(4);
3 System.out.println((char)('a' + (Math.abs(gerador.nextInt()) % 26)));
```

5. **Álgebra Booleana** - Crie um método **iterativo** que recebe uma string contendo uma expressão booleana e o valor de suas entradas e retorna um booleano indicando se a expressão é verdadeira

ou falsa. Cada string de entrada é composta por um número inteiro n indicando o número de entradas da expressão booleana corrente. Em seguida, a string contém n valores binários (um para cada entrada) e a expressão booleana. Na saída padrão, para cada linha de entrada, escreva uma linha de saída com SIM / NÃO indicando se a expressão corrente é verdadeira ou falsa.

6. **Is** - Crie um método **iterativo** que recebe uma string e retorna true se a mesma é composta somente por vogais. Crie outro método **iterativo** que recebe uma string e retorna true se a mesma é composta somente por consoantes. Crie um terceiro método **iterativo** que recebe uma string e retorna true se a mesma corresponde a um número inteiro. Crie um quarto método **iterativo** que recebe uma string e retorna true se a mesma corresponde a um número real. Na saída padrão, para cada linha de entrada, escreva outra de saída da seguinte forma X1 X2 X3 X4 onde cada Xi é um booleano indicando se a é entrada é: composta somente por vogais (X1); composta somente somente por consoantes (X2); um número inteiro (X3); um número real (X4). Se Xi for verdadeiro, seu valor será SIM, caso contrário, NÃO.

7. **Leitura de Página HTML** - Leia duas strings sendo que a primeira é o nome de uma página web e a segunda, seu endereço. Por exemplo, "Pontifícia Universidade Católica de Minas Gerais" e "www.pucminas.br". Em seguida, mostre na tela o número de vogais (sem e com acento), consoantes e dos padrões "$< br >$" e "$< table >$" que aparecem no código dessa página. A entrada padrão é composta por várias linhas. Cada uma contém várias strings sendo que a primeira é um endereço web e as demais o nome dessa página web. A última linha da entrada padrão contém a palavra "FIM". A saída padrão contém várias linhas sendo que cada uma apresenta o número de ocorrência (valor x_i entre parênteses) de cada caractere ou string solicitado. Cada linha de saída será da seguinte forma: a(x_1) e(x_2) i(x_3) o(x_4) u(x_5) á(x_6) é(x_7) í(x_8) ó(x_9) ú(x_{10}) à(x_{11}) è(x_{12}) ì(x_{13}) ò(x_{14}) ù(x_{15}) ã(x_{16}) õ(x_{17}) â(x_{19}) ê(x_{19}) î(x_{20}) ô(x_{21}) û(x_{22}) consoante(x_{23}) $< br >$(x_{24}) $< table >$(x_{25}) nomepágina(x_{26}).

8. **Arquivo em Java**: Faça um programa que leia um número inteiro n indicando o número de valores reais que devem ser lidos e salvos sequencialmente em um arquivo texto. Após a leitura dos valores, devemos fechar o arquivo. Em seguida, reabri-lo e fazer a leitura de trás para frente usando os métodos getFilePointer e seek da classe RandomAccessFile e mostre todos os valores lidos na tela. Nessa questão, você não pode usar, arrays, strings ou qualquer estrutura de dados. A entrada padrão é composta por um número inteiro n e mais n números reais. A saída padrão corresponde a n números reais mostrados um por linha de saída.

9. **Arquivo em C**: Refaça a questão anterior na linguagem C.

10. **RECURSIVO - Palíndromo** - Refaça a questão **Palíndromo** de forma recursiva.

11. **RECURSIVO - Palíndromo em C** - Refaça a questão anterior na linguagem C

12. **RECURSIVO - Ciframento de César** - Refaça a questão **Ciframento de César** de forma recursiva.

13. **RECURSIVO - Alteração Aleatória em C** - Refaça a questão **Alteração Aleatória** de forma recursiva.

14. **RECURSIVO - Álgebra Booleana** - Refaça a questão **Álgebra Booleana** de forma recursiva.

15. **RECURSIVO - Is** - Refaça a questão **Is** de forma recursiva.

4 Exemplo de Resposta em Java

```java
class TP01Q00Aquecimento {
   public static boolean isMaiuscula (char c){
      return (c >= 'A' && c <= 'Z');
   }

   public static boolean isFim(String s){
      return (s.length() >= 3 && s.charAt(0) == 'F' &&
      s.charAt(1) == 'I' && s.charAt(2) == 'M');
   }

   public static int contarLetrasMaiusculas (String s){
      int resp = 0;
      for(int i = 0; i < s.length(); i++){
         if(isMaiuscula(s.charAt(i)) == true){
            resp ++;
         }
      }
      return resp;
   }

   public static void main (String[] args){
      String[] entrada = new String[1000];
      int numEntrada = 0;

      //Leitura da entrada padrao
      do {
         entrada[numEntrada] = MyIO.readLine();
      }
      while (isFim(entrada[numEntrada++]) == false);
      numEntrada--;    //Desconsiderar ultima linha contendo a palavra FIM

      for(int i = 0; i < numEntrada; i++){
         MyIO.println(contarLetrasMaiusculas(entrada[i]));
      }
   }
}
```

5 Exemplo de Resposta em C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define bool      short
#define true      1
#define false     0
#define equals(a, b) (((strcmp(a, b) == 0) ? true : false))
#define NUMENTRADA 1000
#define TAMLINHA   1000

bool isMaiuscula (char c){
    return (c >= 'A' && c <= 'Z');
}

bool isFim(char* s){
    return (strlen(s) >= 3 && s[0] == 'F' &&
    s[1] == 'I' && s[2] == 'M');
}

int contarLetrasMaiusculas (char* s){
    int resp = 0;

    for(int i = 0; i < strlen(s); i++){
        if(isMaiuscula(s[i]) == true){
            resp ++;
        }
    }

    return resp;
}

int main(int argc, char** argv){
    char entrada[NUMENTRADA][TAMLINHA];
    int numEntrada = 0;

    //Leitura da entrada padrao
    do {
        fgets(entrada[numEntrada], TAMLINHA, stdin);
    }
    while (isFim(entrada[numEntrada++]) == false);
    numEntrada--;    //Desconsiderar ultima linha contendo a palavra FIM

    for(int i = 0; i < numEntrada; i++){
        printf("%i\n",contarLetrasMaiusculas(entrada[i]));
    }
}
```