Nome_____ Matrícula_____

LISTA DE EXERCÍCIOS 04

Implemente o que se pede abaixo na forma de funções armazenadas em arquivos-fonte que contenham apenas elas e elementos de apoio, como funções auxiliares, e arquivos de cabeçalho onde estruturas de dados e assinaturas das funções são criadas, em C++. Crie também programas que as utilizem e demonstrem seus funcionamentos:

1. Crie uma função que receba o nome de um arquivo texto e retorne o número de linhas, quantidade de caracteres (exceto os de controle) e o número de palavras. O programa que a testa deve receber o nome do arquivo pela linha de comando e imprimir essa informação na saída padrão. Ele deve ser invocado como segue (2,0):

```
$ conta arquivo.txt
```

2. Crie uma função que compare dois arquivos binários cujos nomes são passado como parâmetro, imprimindo cada posição em que diferem e qual é o byte de cada um naquelas posições. O programa que a testa deve receber o nome dos dois arquivos pela linha de comando. Caso os arquivos sejam binariamente idênticos, deve ser mostrada uma mensagem correspondente. Ele deve ser invocado como segue (1,5):

```
$ compare a.txt b.txt
```

3. Implemente uma função de busca em um arquivo texto, que recebe como parâmetros o nome do arquivo e o texto a ser buscado. A função deve imprimir as linhas em que o texto aparece com seu número como prefixo. O programa de teste receberá o nome do arquivo e o texto pela linha de comando, nessa ordem (1,5):

```
$ busque texto arquivo
```

4. Crie um programa que receba nomes de arquivos pela linha de comando e use uma função que concatene todos os arquivos passados no último arquivo. Observe que este não existe, necessariamente, sendo preenchido com as linhas dos demais, na ordem em que estes aparecem na linha de comando, como por exemplo: (1,5)

```
$ concatene a.txt b.txt c.txt d.txt resultado.txt
```

5. Crie um programa que receba o nome de três arquivos pela linha de comando. O primeiro deve ser um arquivo html com marcadores de templates no formato `${xx}`, onde `xx` é um número hexadecimal. O segundo é um arquivo texto cujas linhas tem o formato `xx=valor`, onde `xx` é o hexadecimal que identifica um parâmetro e valor é um texto correspondente a esse valor. O programa deve varrer o arquivo html e substituir todos os templates pelos seus respectivos valores. Na eventualidade de um dos templates fazer referência a um valor não existente no segundo arquivo, deve-se substituí-lo pelo texto `!!!` . Essa substituição deve ser gravada no arquivo cujo nome é o terceiro parâmetro passado na linha de comando. O programa não deve manter mais que uma linha do arquivo html por vez, fazendo a substituição e gravando-a imediatamente. No final, deve apresentar na tela a quantidade de substituições de cada parâmetro no arquivo. O formato de linha de comando seria (3,5):

```
$ processe template.html parametros.txt resultado.html
```