Habilidades BÁSICAS de processamento de strings

HABILIDADE 1:

Dado um arquivo de texto que contenha apenas caracteres alfabéticos `[A-Za-z]`, dígitos `[0-9]`, espaços e pontos (“ `.` ”), escreva um programa para ler esse arquivo linha por linha até que seja encontrado uma linha que começa com sete pontos (“`.......`”). Concatene (combine) cada linha em uma grande string T. Quando duas linhas são combinadas, coloque um espaço entre elas para que a última palavra da linha anterior esteja separada da primeira palavra da próxima linha (atual). Existirão até 30 caracteres por linha e não mais do que 10 linhas nesse arquivo de texto. Não existirão espaços sobrando no final das linhas e cada linha será terminada por um caractere newline (“`\n`”). Um exemplo de um arquivo que seu programa deve ser capaz de processar é o seguinte:

```
I love CS101 Competitive
Programming. i also love
AlGoRiThM
.......you must stop after reading this line as it starts with 7 dots
after the first input block, there will be one looooooooooooog line...
```

Agora responda as seguintes perguntas e tenha certeza de que você domina tudo o que está sendo perguntado (se você não dominar, estude!!!):

a) Você sabe como armazenar uma string corretamente na linguagem que está usando?
b) Como ler um texto de input linha por linha?
c) Como concatenar (combinar) duas string em uma string maior?
d) Como verificar linhas que começam “ `.......` ” para interromper a leitura do input?

HABILIDADE 2:

Suponha que você tenha uma grande string T. Nós queremos checar se uma outra string P pode ser encontrada em T. Imprima todos os índices onde P aparece em T ou imprima “ `-1` ” se P não estiver contida em T. Por exemplo,

se T = “`I love CS101 Competitive Programming. i also love AlGoRiThM`”
se P = “`I`”

então o output será apenas `{0}` (índices começam em zero) pois “`I`” e “`i`” são caracteres diferentes em ASCII. Outros exemplos:

se P = “`love`”, então o output será `{2, 45}`;
se P = “`book`”, então o outupt será `{-1}`.

Agora responda as seguintes perguntas e tenha certeza de que você domina tudo o que está sendo perguntado (se você não dominar, estude!!!):

a) Como encontrar a primeira ocorrência de uma substring dentro de uma string (se existir)? Será preciso implementar um algoritmo de emparelhamento de strings (como o Knuth-Morris-Pat) ou podemos utilizar as bibliotecas da linguagem?
b) Como encontrar a próxima ocorrência de uma substring em uma string (se existir)?

HABILIDADE 3:

Suponha que você precise fazer alguma análise simples dos caracteres em uma string T e também transformar todos os caracteres de T em minúsculas. A análise necessária é: a) Quantos dígitos `[0-9]`, quantas vogais `[aeiouAEIOU]` e consoantes `[todas as não-vogais, maiúsculas ou minúsculas]` existem em T? Como você consegue fazer isso em O(n) onde n = comprimento da string?

HABILIDADE 4:

Nós precisamos quebrar uma long string T em *tokens* (substrings) e armazenar cada um desses *tokens* em um array de strings chamado `tokens`. Para esta tarefa, os delimitadores desses *tokens* são espaços e pontos (assim, quebrando sentenças em palavras).

Por exemplo, se nós "tokenizarmos" a string T (em minúsculas) mostrada na "Habilidade 2", teríamos os seguintes *tokens*:

```
{"i", "love", "cs101", "competitive", "programming", "i", "also", "love" "algorithm"}
```

Agora precisamos ordenar esse array de modo lexicográfico (basicamente a ordem do dicionário) e, então, encontrar a menor string lexicográfica. Após a ordenação dos tokens teremos:

```
{"algorithm", "also", "competitive", "cs101", "i", "i", "love", "love", "programming"}
```

A menor palavra lexicográfica é então "`algorithm`", pois está no começo da ordem. Crie um programa que faça exatamente o que esta habilidade está demonstrando. Depois responda as seguintes perguntas e tenha certeza de que você domina tudo o que está sendo perguntado (se você não dominar, estude!!!):

a) Como tokenizar uma string?
b) Como armazenar os *tokens* (as string menores) em um array de strings?
c) Como ordenar lexicograficamente um array de strings?

HABILIDADE 5:

Agora identifique qual palavra aparece com maior freqüência na string T. Para fazer isso precisamos contar a freqüência de cada palavra. Para T, a resposta é "`i`" ou "`love`", pois elas aparecem duas vezes. Que estrutura de dados deveria ser utilizada para isso?

HABILIDADE 6:

O arquivo texto de exemplo da "Habilidade 1" tem uma linha a mais após a linha que começa com "`.......`" e o comprimento dessa linha adicional não está dentro dos limites impostos pelo problema. Crie um programa para contar quantos caracteres existem nessa última linha. Como ler uma string se o comprimento não é conhecido previamente?