UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE - CTS
DEPARTAMENTO DE COMPUTAÇÃO – DEC

DISCIPLINA: LINGUAGENS DE PROGRAMAÇÃO 2
PROFESSOR ANTONIO CARLOS SOBIERANSKI
a.sobieranski@ufsc.br

ENUNCIADO TRABALHO T3 – CIPHER

A ser desenvolvido em: DUPLAS

O trabalho T3 consiste na implementação de um cifrador de texto, onde a estratégia para cifrar um manuscrito de entrada é a sua própria frequência dos caracteres. Desta forma, significa dizer que a cifragem será única para cada manuscrito a ser utilizado como *input*.

Não há menu para o programa T3, porém as entradas serão definidas em sequência:

```
cout << "enter with an alphabet to read: ";
string pathAlpha;
cin >> pathAlpha;
.....

cout << "enter with a manuscript to encode: ";
string pathManusc;
cin >> pathManusc;
....
```

Logo, são utilizadas somente 2 entradas para o programa:

- Arquivo de alfabeto, correspondendo a um simples arquivo de texto contendo uma sequência de símbolos da tabela ASCII, que logo mais será **associado** aos caracteres do manuscrito
- Arquivo contendo um texto a ser cifrado, chamado de manuscrito. A frequência de palavras do manuscrito determinará a **associação** com o alfabeto.

Por exemplo:

Alfabeto exemplo	**Frequência de caracteres do manuscrito em ordem decrescente**
0	‘a’ (contém 35 caracteres)
1	‘e’ (contém 26 caracteres)
2	‘c’ (contém 15 caracteres)
3	‘b’ (contém 07 caracteres)
4	‘i’ (contém 06 caracteres)
5	….
6	
7	
8	
9	
?	
@	
A	
B	
C	
….	

A saída final do algoritmo é o texto final cifrado, ou seja, considerando a tabela acima, substituir do manuscrito todos os ‘a’ por ‘0’, ‘e’ por ‘1’, ‘c’ por ‘2’, e assim sucessivamente, caractere a caractere do manuscrito de entrada.

Input: *alpha1.txt e manuscript1.txt*

Output:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercita tion ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Except eur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.	;&)#-!"/,'-!+&*&)!,"$!%-#$1!.&(,#. $#$')!%+"/","."(5!#*"$1!,#+!+&!#"',-&+! $#-/&)!"(."+"+'($!'$!*%3&)#!#$!+&*&)#!- %5(%!%*"0'%2!<$!#("-!%+!-"("-!6#("%-1! 0'",!(&,$)'+!#7#)."$% $"&(!'**%-.&!*%3&)",!(","!'$!%*"0'"/! #7!#%!.&--&+&!.&(,#0'%$2!9'",! %'$#!")')#!+&*&)!"(!)#/)#=#(+#)"$!"(! 6&*'/$%$#!6#*"$!#,,#!."**'-!+&*&)#!#'! 4'5"%$!('**%!/%)"%$')2!:7.#/$ #')!,"($!&..%#.%$!.'/"+%$%$! (&(!/)&"+#($1!,'($!"(!.'*/%!0'"! &44"."%!+#,#)'($!-&**"$!%("-!"+!#,$!* %3&)'-28

Obs.:

1. considerar nos arquivos de manuscrito o caractere ‘\n’.

2. caso o alfabeto lido não comporte a quantidade de caracteres necessária em um arquivo de manuscrito, a mensagem de erro deve ser mostrada: “Error, alphabet with insufficient characters to encode the manuscript”

3. Usar stable_sort para garantir a ordem correta dos pares – alguns caracteres podem possuir a mesma frequência de caracteres.

Dicas:

1. os valores da tabela ASCII podem ser utilizados como índices do vetor
2. a associação pode ser feita por **std::pair** ou **std::map**
3. o alfabeto pode ser lido tanto por *getline*, como por *get* (já que o alfabeto possuirá um caractere somente, por linha)
4. o manuscrito todo pode ser lido caractere a caractere, ou linha a linha, lembrando que um texto pode conter parágrafos. Pode ser usado **std::stringstream** para tal, que neste caso é mais versátil que um simples **std::string**.

Alguém disse novamente em ‘sabedorias’: Debug salva vidas….