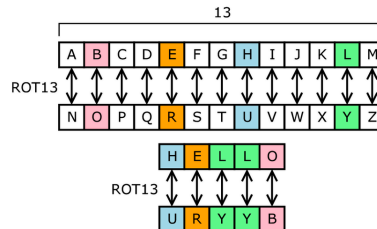


```
-- Primeira Prova de Paradigmas de Linguagens Computacionais
-- 1/2016 - 05/05/2016
--
```

ROT-13 é o nome que se costuma usar para um procedimento simples mas eficaz para garantir que textos eletrônicos não sejam lidos por distração ou acidente. ROT-13 vem do inglês, ROTate by 13 places, "ROTacionar 13 posições".

ROT-13 é uma cifra de César aplicável apenas aos caracteres alfabéticos (da língua inglesa) e com passo 13.

o algoritmo faz a troca conforme o exemplo abaixo:



**1) (2.0)** Considerando uma chave para representar esta cifra como uma lista de pares informando que caracter deve substituir qual outro, implemente uma função cipher para realizar esta substituição em uma String, a partir de uma Chave, retornando uma String modificada como resultado.

Qualquer caracter que não esteja na Chave deve aparecer inalterado na String de saída.

```
type Chave = [(Char, Char)]
```

```
rot13parcial :: Chave -- troca 'a' por 'n', 'b' por 'o' etc.
rot13parcial = [('a','n'),('b','o'),('c','p'),('d','q'),('e','r'),('f','s'),
                ('g','t'),('h','u'),('i','v'),('j','w'),('k','x'),('l','y'),('m','z')]
```

```
-- cipher :: Chave -> String -> String
-- exemplo: cipher rot13parcial "hello*hello" ----> "uryyo*uryyo"
```

**2) (2.0)** Para decifrar o código é preciso inverter a Chave. Faça uma função que inverta a ordem de cada um dos pares de uma Chave (lista de pares de caracteres):

```
-- inverteChave :: Chave -> Chave
-- exemplo: inverteChave rot13parcial ----> [('n','a'),('o','b'),('p','c'),...
```

**3) (2.0)** Uma implementação diferente da função cipher poderia usar uma função no lugar dos pares de caracteres para representar a chave.

Defina esta outra versão da função cipher, chamada cipherf

```
type FuncaoChave = (Char -> Char)
```

```
trocaSoLetraL :: FuncaoChave
trocaSoLetraL 'l' = 'b'
trocaSoLetraL c   = c
```

```
-- cipherf :: FuncaoChave -> String -> String
-- exemplo: cipherf trocaSoLetraL "hello*hello" ----> "hebbo*hebbo"
```

**4) (2.0)** usando uma expressão do tipo Chave (lista de pares), você conseguiria gerar uma função que pode ser usada como chave para a função cipherf?

Em caso afirmativo, demonstre. Caso contrário, justifique.

```
chaveToFuncaoChave :: Chave -> FuncaoChave
-- exemplo: cipherf (chaveToFuncaoChave rot13parcial) "hello*hello" ----> "uryyo*uryyo"
```

**5) (2.0)** Para uma maior eficiência no processo de busca dos caracteres em uma chave, podemos fazer a busca dos caracteres a serem trocados usando uma árvore binária em que cada nó, além da informação de uma letra e sua substituição, possua sub-árvores para a busca de letras menores ou maiores que aquela letra. Se a letra não for achada na árvore a substituição não ocorre. Escreva esta função cipherT, que usa uma árvore:

```
data KeyTree = Node Char Char KeyTree KeyTree | Empty
chaveParcial :: KeyTree
chaveParcial = Node 'h' 'u' (Node 'c' 'p' (Node 'b' 'o' (Node 'a' 'n' Empty Empty) Empty)
                                (Node 'e' 'r' Empty Empty))
                                (Node 'l' 'y' Empty (Node 'm' 'z' Empty Empty))
-- cipherT :: KeyTree -> String -> String
-- exemplo: cipherT chaveParcial "hello*hello" ----> "uryyo*uryyo"
```