# Computação 1, 2020.1
# Lista 6

Para entrega até 1/2 às 10:00

Submeta suas soluções colocando os arquivos correspondentes na sua pasta do Google Drive*

Lembre-se de escolher bons nomes para suas funções e variáveis, de documentar seu código com *docstrings* (documentação de função) e comentários onde for apropriado, e de fazer testes para suas funções (na documentação usando o módulo `doctest` ou em funções dedicadas).

**Questão 1.** De acordo com o que se relata, o general e ditador romano Julio César usava o seguinte sistema, hoje conhecido como *Cifra de César*, para enviar mensagens secretas para seus aliados:

- utilizando um certo alfabeto, pré-combinado, e
- utilizando um certo número natural $x > 0$, pré-combinado,
- cada símbolo da mensagem (usando apenas símbolos no alfabeto pré-combinado) era trocado pelo símbolo $x$ posições à direita no alfabeto (se o final do alfabeto fosse atingido, voltava-se ao início).

Por exemplo, com alfabeto `ABCDEFGHI` e $x = 5$, a palavra `CIDADE`, quando codificada, virava `HEIFIA`.

**a.** Faça uma função que implemente a codificação da cifra de César, recebendo como entradas a `mensagem` a ser codificada, o `alfabeto` (como uma única `str` contendo todos os símbolos, como no exemplo acima) e o valor de `x` a ser usado, e retornando a codificação correspondente da `mensagem`.

**b.** Faça uma função que implemente a decodificação da cifra de César, recebendo como entradas a `mensagem` codificada, o `alfabeto` e o valor de `x` que foi usado na codificação, e retornando a versão decodificada da `mensagem`.

**Questão 2.** No departamento de recursos humanos de uma certa empresa, os dados dos(as) funcionários(as) são guardados em um grande arquivo de texto. Cada linha do arquivo corresponde a um(a) funcionário(a), de acordo com a seguinte regra:

1. Do começo da linha até o símbolo `#`, a string contém um número correspondente ao cargo do(a) funcionário(a);

*Link recebido por email em 7/12/2020 — o nome é parecido com `<seu nome> - Computação 1 - Submissões e Feedback`.

2. após o símbolo `#` e até o símbolo `@`, a string contém o nome do(a) funcionário(a);

3. após o símbolo `@` e até o fim da string ou o símbolo `&` (o que vier primeiro), a string contém a data de nascimento do(a) funcionário(a) no formato `DDMMAAAA`;

4. opcionalmente, após o símbolo `&` e até o fim da string ou o símbolo `%` (o que vier primeiro), a string contém o endereço do(a) funcionário(a);

5. opcionalmente, após o símbolo `%` e até o fim da string, a string contém o tipo sanguíneo do(a) funcionário(a).

As informações de cada funcionário(a) sempre aparecem na ordem listada acima, e o tipo sanguíneo do(a) funcionário(a) só está registrado no arquivo caso o endereço também esteja.

Faça uma função que receba uma string, correspondendo a uma das linhas do arquivo do RH da empresa que acabamos de descrever, e retorne, em ordem:

- um `int` correspondendo ao cargo do(a) funcionário(a) em questão;
- uma `str` correspondendo ao nome do(a) funcionário(a);
- um `int` correspondendo ao dia de nascimento do(a) funcionário(a);
- um `int` correspondendo ao mês de nascimento do(a) funcionário(a);
- um `int` correpondendo ao ano de nascimento do(a) funcionário(a);
- uma `str` correspondendo ao endereço do(a) funcionário(a), caso esta informação esteja presente, ou `None` em caso contrário; e
- uma `str` correspondendo ao tipo sanguíneo do(a) funcionário(a), caso esta informação esteja presente, ou `None` em caso contrário.

**Questão 3.** Em um alfabeto com apenas dois símbolos, `+` e `-`, digamos que uma expressão é *bem formada* se:

1. as quantidades de `+`s e `-`s na expressão são iguais;

2. à esquerda de cada símbolo `-`, a quantidade de `+`s é estritamente maior que a quantidade de `-`s.

Em outras palavras, uma expressão é bem formada sse começando com 0 e interpretando cada `+` como $+1$ e cada `-` como $-1$, chegamos ao final da expressão novamente com 0 e sem termos passado por números negativos em nenhum momento.

Faça uma função que receba uma string correspondendo a uma expressão neste alfabeto e retorne um booleano indicando se a expressão é bem formada ou não.

**Questão 4.** Em um alfabeto com apenas dois símbolos, `)` e `(`, digamos que uma expressão é *bem formada* se ela pode ser obtida de alguma expressão aritmética bem formada (de acordo com as regras aprendidas no ensino fundamental) apagando-se todos os símbolos que não são `)` nem `(`. Por exemplo, a expressão `(()(()))` é bem formada, pois poderia ter vindo da expressão aritmética `((2+3)/((1+1)*2))`, mas a expressão `(()` não é bem formada.

**a.** Dê a melhor caracterização que você puder para as expressões bem formadas nesta linguagem, ou seja, encontre a propriedade mais "interessante" que você conseguir de forma que uma expressão com os símbolos `)` e `(` é bem formada sse ela tem a propriedade dada.

**b.** Faça uma função que receba uma string correspondendo a uma expressão neste alfabeto e retorne um booleano indicando se a expressão é bem formada ou não.

# Desafio (opcional)

**Questão 5.** Sobre a Questão 1: como vimos, na Cifra de César, cada letra na posição $k$ é substituída pela letra na posição $k + x$.

**a.** Faça uma função para implementar a codificação da versão *multiplicativa* da Cifra de César, onde cada letra na posição $k$ é substituída pela letra na posição $k \cdot x$ (voltando ao início do alfabeto sempre que necessário).

**b.** Mostre que nem toda mensagem de todo alfabeto com toda chave de multiplicação pode ser decodificada. Na verdade, para cada alfabeto, existe pelo menos um $x \neq 0$ tal que mensagens daquele alfabeto codificadas com multiplicação por $x$ não podem ser decodificadas.

**c.** Faça uma função para implementar a decodificação da versão multiplicativa da cifra de César *sempre que possível*: a função deve retornar a mensagem decodificada, se essa mensagem puder ser determinada, ou `False` em caso contrário.

**Questão 6.** Sobre a Questão 4.

**a.** Dê uma caracterização recursiva das expressões bem formadas e implemente uma função que testa se uma expressão é bem formada, usando a sua caracterização recursiva.

**b.** Faça uma função que receba uma expressão bem formada `expr` e um natural `n` e retorne a menor subexpressão bem formada de `expr` que contenha (o símbolo na) a posição `n` da `expr` (começando a contagem de `0`, como sempre).

Por exemplo, com entradas `()(()(()()))` e `1`, a saída deve ser `()`, e com entradas `()(()(()()))` e `5`, a saída deve ser `(()())`.