Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES

Programação Funcional

Prof. Oberlan Romão

Lista de exercícios 5

1. Faça uma função que receba duas listas (`l1` e `l2`) e retorne uma lista com a união de `l1` e `l2` de forma que que a nova lista não contenha elementos repetidos. Por exemplo, se `l1 = [1, 2, 3, 3]` e `l2 = [1, 5, 3]`, a união de `l1` e `l2` deve ser `[1, 2, 3, 5]` (a ordem dos elementos na lista não importa). Dica: use o operador `in`;

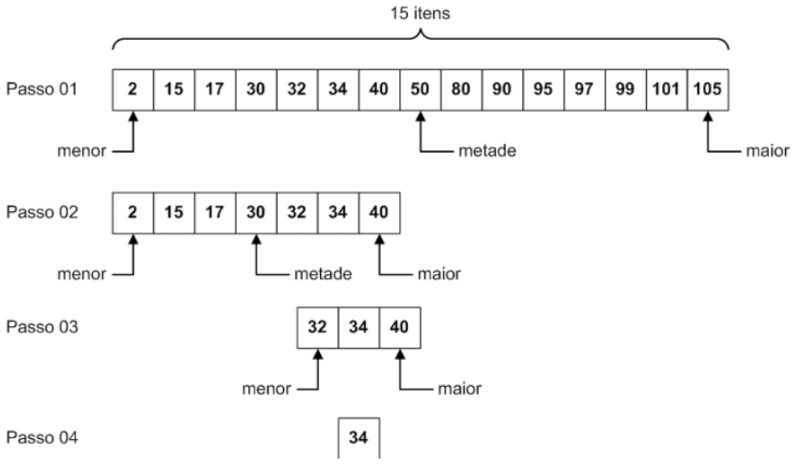
2. Defina uma função que receba uma lista `l1` e retorne outra lista de forma que elementos repetidos em `l1` apareça apenas uma vez na nova lista. Por exemplo, se `l1 = [1, 2, 3, 3, 2]`, sua função deve retornar `[1, 2, 3]`.

3. Faça uma função que receba duas listas e retorne uma lista com a interseção dessas listas. Por exemplo, se `l1 = [1, 2, 3, 3]` e `l2 = [1, 5, 3]`, a interseção de `l1` e `l2` deve ser `[1, 3]`. Note que a lista resultante não pode conter elementos repetidos;

4. Faça uma função que receba uma lista e retorna `True` se a lista estiver ordenada de forma crescente e `False`, caso contrário.

5. Um algoritmo de busca muito utilizado quando os elementos de uma lista estão ordenados é a *Busca Binária*. A ideia do algoritmo é testar o elemento procurado com o valor do elemento armazenado no meio da lista. Se o elemento buscado for menor que o elemento do meio, pode-se concluir que, se o elemento estiver presente no vetor, ele estará na primeira parte do vetor; se for maior, estará na segunda parte do vetor; se for igual, o elemento no vetor foi encontrado. Se for concluído que o elemento está em uma das partes da lista, o procedimento é repetido considerando apenas a parte que restou: compara-se o elemento pesquisado com o elemento armazenado no meio dessa parte. Este procedimento é continuamente repetido, subdividindo a parte de interesse, até o elemento ser encontrado ou se chegar a uma parte da lista com tamanho zero.

   A figura abaixo exemplifica a busca binária em uma lista ordenada. Neste caso, está sendo pesquisado o elemento 34.



Implemente uma função (buscaBinariaR) que faça uma Busca Binária recursiva em uma lista . Se o elemento procurado (x) estiver na lista, a função deve retornar a posição onde ele se encontra, caso contrário, ela deve retornar None. Sua função deve obedecer ao cabeçalho apresentado abaixo.

```
def buscaBinaria(L, x):
  '''Função utilizada apenas para facilitar a chamada da função buscaBinariaR'''
  return buscaBinariaR(L, x, 0, len(L)-1)

def buscaBinariaR(L, x, ini, fim):
  #implemente aqui
```

6. Faça um programa que peça para o usuário digitar valores inteiros entre 0 e 9, uma entrada fora dessa faixa indica o fim da leitura. Em seguida, seu programa deve imprimir a quantidade de números digitados e a frequência (quantidade de vezes) que cada número foi digitado. Veja um exemplo.

```
Digite os valores (um número <0 ou >9 indica o fim da leitura):
1 1 1 5 1 0 0 3 2 5 5 1 5 6 7 -1

Numeros digitados: 15
Frequencia de cada numero
0: 2
1: 5
2: 1
3: 1
4: 0
5: 4
6: 1
7: 1
8: 0
9: 0
```

7. Elabore um programa que peça ao usuário para inserir a temperatura dos últimos n dias ($1 \leq n \leq 100$) e calcule a temperatura média (m) e o desvio padrão (dp) considerando as informações inseridas pelo usuário. Seu programa deve, obrigatoriamente, ter três funções (com exceção da `main()`): uma para fazer a leitura das temperaturas, uma para calcular a média e outra para calcular o desvio padrão. A primeira recebe o valor de n e faz a leitura dos dados, a segunda recebe uma lista com as temperaturas e retorna a média dos valores, a terceira calcula e retorna o desvio padrão.

   Obs.: Supondo que a média de um conjunto de n valores seja m, o desvio padrão é dado pela expressão abaixo (onde t_i é o valor de temperatura no dia i).

$$dp = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - m)^2}$$

8. Faça um programa que contenha duas função: `ehVogal` e `contaVogal`. A primeira recebe um caractere e retorna `True` se o caractere é uma vogal e `False`, caso contrário. A segunda função, (`contaVogal`), receba uma string e retorne a quantidade de vogais contidos na string, ela deve obrigatoriamente usar a função `ehVogal`. Implemente, também, uma função `main()` que lê uma string e utiliza a sua função `contaVogal` para imprimir o número de vogais na string.

9. Considerando as funções do exercício anterior e supondo que uma string contenha apenas letras (minúsculas ou minúsculas), implemente uma função `contaConsoante`, usando a função `ehVogal`, para contar o número de consoantes. Implemente uma segunda versão usando a função `contaVogal`.

10. Em Python, assim como em várias outras linguagens, um caractere é armazenado internamente em um byte e são codificados em números usando a codificação Unicode (https://en.wikipedia.org/wiki/List_of_Unicode_characters). Por exemplo, o dígito 'A' é representado pelo número decimal 65, o 'B' por 66 e assim por diante. Podemos usar a função integrada `ord` para descobrir o código Unicode de um caractere. Já a função `chr`, faz o contrário, imprime o caractere representado pelo valor inteiro. Por exemplo,

```
>>> ord('A')
65
>>> chr(65)
'A'
>>> ord('a')
97
```

Usando essas duas função imprima as letras maiúsculas do alfabeto.

11. Faça um programa que contenha três função: `ehLetra`, `ehMaiuscula`, `converteMaiuscula`. A função `ehLetra` recebe um caractere e returna `True` se o caractere for uma letra (maiúscula ou minúscula) e `False`, caso contrário. A função `ehMaiuscula` recebe um caractere e returna `True` se o caractere for uma letra maiúscula e `False`, caso contrário. Já a função `converteMaiuscula` recebe uma string e, usando as função anteriores, converte as letras da string em maiúscula. Por exemplo,

```
>>> s = "teste 123"
>>> print(converteMaiuscula(s))
TESTE 123
```

12. Como já sabemos, a função `map` é uma função integrada de Python utilizada para aplicarmos uma função a cada elemento de uma lista, retornando uma nova lista contendo os elementos resultantes da aplicação da função. Por exemplo:

```
>>> import math
>>> lista1 = [1, 4, 9, 16, 25]
>>> lista2 = list(map(math.sqrt, lista1)) #temos que transformar o retorno para
                                          lista usando a função integrada list.
>>> print(lista2)
[1.0, 2.0, 3.0, 4.0, 5.0]
>>> lista3 = list(map(lambda x: x**2, lista2))
>>> print(lista3)
[1.0, 4.0, 9.0, 16.0, 25.0]
```

Defina uma função, chamada `myMap`, que receba uma função `f` e uma lista e retorna uma nova lista com os elementos modificados pela função `f`.

13. Outra função integrada é a `filter`. Como o próprio nome já diz, ela filtra os elementos de uma sequência, "deixando passar" para a sequência resultante apenas os elementos para os quais a chamada da função que o usuário passou retornar `True`. Por exemplo,

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> p = list(filter(lambda x : x % 2 == 0, a))
>>> print(p)
[2, 4, 6]
```

Considerando a lista: `l1 = list(range(2, 101))` e usando a função `filter`, crie uma lista `l2` que contenha apenas:

a. Números divisíveis por 3 e 5;

b. Números primos.

14. Usando compreensão de lista, crie uma lista que represente os seguintes conjuntos:

a. $S_1 = \{x^2 + 2 \mid x \in \mathbb{N}, x \leq 10\}$;

b. $S_2 = \{x \mid x \in \mathbb{Z}, -100 \leq x \leq 100, x \text{ é divisível por 3 e 5}\}$;

c. $S_3 = \{x \mid x \in \mathbb{N}, x \leq 500, x \text{ é um número perfeito}\}$. Você deve implementar uma função para verificar se x é perfeito ou não;

d. $S_4 = \{x \mid x \in \mathbb{N}, x \leq 100, x \text{ é um número primo}\}$. Você deve implementar uma função para verificar se um x é primo ou não;

e. $S_5 = \{F_n \mid n \in \mathbb{N}, 1 \leq n \leq 10\}$, onde F_n representa o n-ésimo termo da sequência de Fibonacci.