Computação I

Lista de exercícios 3 – A estrutura de repetição while e chamada de função dentro de função.
Atenção! Leia as instruções antes de fazer a lista! A padronização do nome do arquivo e dos nomes das funções é muito importante e está explicada no arquivo de instruções!
Data de entrega: 05/10/2022
A menos que esteja explicitamente pedido na questão, não utilize nenhum método ou função já existente do Python exceto pela função print. De forma simplificada, função é tudo o que precisa ser chamado com parênteses, e método é tudo o que precisa ser chamado com parênteses, mas que está associado a uma variável, isto é, variavel.metodo(args). Funções de transformação de tipo (int, float, str) também não são permitidas. Não importe nenhum módulo. Não é necessário testar se os dados passados por argumento são válidos. Nessa lista, utilize somente a estrutura de repetição while e **não utilize o for e não utilize recursão**. Não crie funções dentro de funções.

1. Escreva uma função em Python que recebe, por argumento, dois números inteiros e positivos, x e y, e retorna o resultado de x multiplicado por y, mas sem utilizar o asterisco para fazer a multiplicação (Dica: é possível definir a multiplicação através de um somatório?).

2. Considere uma sequência definida por todos os números inteiros entre 1 e um número n qualquer maior do que 1. Dentro desta sequência, podemos considerar que cada número inteiro define, também, uma subsequência, que vai de 1 até ele próprio. Por exemplo: a sequência de 1 a 4 contém quatro subsequências: 1, 1 a 2, 1 a 3, e 1 a 4 (sendo esta última a própria sequência original). Escreva uma função em Python que receba um número n maior do que 1, e retorne a soma total da soma de todos os termos presentes em cada subsequência. Por exemplo: para entrada (4), a saída deve ser 20. Esse valor é obtido da seguinte forma: (1) + (1 + 2) + (1 + 2 + 3) + (1 + 2 + 3 + 4). No cálculo anterior, entre parênteses estão as somas parciais, as quais consideram cada subsequência presente no intervalo original. O resultado final, a ser retornado pela função, deve ser a soma total dessas somas parciais, obtidas de cada subsequência. Observe que termos que aparecem em mais de uma subsequência são contados mais de uma vez. O número 1, por exemplo, aparece em todas as quatro subsequências, portanto, ele é contabilizado 4 vezes na soma total (o número 2 é contabilizado três vezes porque aparece em três das quatro subsequências, e assim por diante). Utilize obrigatoriamente o loop while nesta questão.

3. A análise combinatória é uma área da matemática muito utilizada para cálculo de probabilidades. Dentro dessa área, é possível calcular quantos grupos diferentes podem ser formados considerando n elementos e considerando que esses elementos são agrupados com k elementos por grupo, onde k é menor ou igual a n. Se a ordem dentro do grupo é importante, então dizemos que queremos calcular o arranjo para n elementos tomados k a k ($A_{n,k}$). Se a ordem não é importante, então dizemos que queremos calcular a combinação para n elementos tomados k a k ($C_{n,k}$). Por exemplo, quantos grupos podemos formar com dois elementos (*a* e *b*) tomados dois a dois? Considerando o arranjo, podemos formar os grupos *ab* e *ba*, ou seja, dois grupos. Considerando a combinação, *ab* seria equivalente a *ba*, pois a ordem não importa. Logo, considerando combinação, somente um grupo pode ser formado. O arranjo e a combinação podem ser calculados da seguinte forma:

$$A_{n,k} = \frac{n!}{(n-k)!} \quad ; \quad C_{n,k} = \frac{n!}{k!(n-k)!} = \frac{1}{k!}A_{n,k}$$

   Nessa questão você deve escrever três funções em Python: a primeira (cujo nome é livre) recebe um número inteiro e maior ou igual a zero por argumento e retorna o seu fatorial; a segunda (cujo nome deve ser questao3a) recebe n e k ($n \geq k$), inteiros e positivos, por argumento (nessa ordem) e retorna

o arranjo de n elementos tomados k a k ($A_{n,k}$); e a terceira (cujo nome deve ser questao3b) recebe n e k ($n \geq k$), inteiros e positivos, por argumento (nessa ordem) e retorna a combinação dos n elementos tomados k a k ($C_{n,k}$). A função para cálculo do arranjo deve chamar a primeira função (aquela cujo nome é livre) para calcular os fatoriais da fórmula. A função para cálculo da combinação, por outro lado, deve chamar tanto a função que calcula o fatorial (para encontrar $k!$), quanto a função que calcula o arranjo (para o outro termo da fórmula). Ou seja, questao3a chama a função que calcula o fatorial; e questao3b chama a função questao3a e a função que calcula o fatorial. Em ambos os casos, os valores retornados através das chamadas das funções devem ser utilizados para calcular o valor de retorno da função que fez a chamada. Obs.: por definição, o fatorial de 0 é igual a 1.

4. Nesta questão você deve escrever duas funções. A primeira função, cujo nome deve terminar com 4a, recebe por argumento um número inteiro e maior que zero, e retorna o booleano True se esse número for um número primo, ou o booleano False se esse número não for um número primo. A segunda função, cujo nome deve terminar com 4b, recebe um número n por argumento, inteiro e maior que zero, e retorna o número primo imediatamente maior que n, ou o próprio n, se n for primo. A função 4b deve chamar a função 4a, e utilizar o valor retornado pela função 4a para descobrir o número que deve retornar. Ambas funções devem usar while. Exemplos de nomes de função válidos para a questão 4a: verificarPrimo4a, primo4a, questao4a... Exemplos de nomes de função válidos para a questão 4b: encontrarProximoPrimo4b, encontrarPrimo4b, questao4b...

5. Considere uma caixa d'água de capacidade de x litros. Uma bomba é acionada para encher a caixa d'água, que se encontra inicialmente vazia. A bomba consegue fornecer, inicialmente, y litros para a caixa d'água ao longo de um dia. Entretanto, a água também é consumida ao longo do dia, de modo que, ao final de cada dia, sobra apenas metade da água disponível na caixa naquele dia. A vantagem é que, no dia seguinte, a bomba consegue fornecer 20% a mais da quantidade de água que forneceu no dia anterior. Faça uma função em Python que receba como argumentos de entrada, e nesta ordem, a capacidade total da caixa d'água e a quantidade de água inicialmente fornecida pela bomba, e que retorne o número de dias necessários para que a caixa esteja completamente preenchida ao final do dia considerando que a caixa está inicialmente vazia. Por exemplo, se os valores de entrada forem (20,15). No dia 1 a bomba fornece 15 litros, resultando, no fim do dia, em 15/2 litros (água fornecida menos o consumo) = 7.5 litros no final do dia. No dia 2 a bomba fornece 15 + 15*0.2 litros = 18 litros. No fim do dia 2 a quantidade de água é de (18+7.5)/2 = 12.75 (fornecimento do dia mais o que sobrou do dia anterior, menos o consumo). No dia 3 a bomba fornece 18 + 18*0.2 litros = 21.60 litros. No fim do dia 3 a quantidade de água é de (21.6+12.75)/2 = 17.175 (fornecimento do dia mais o que sobrou do dia anterior, menos o consumo). No dia 4 a bomba fornece 21.60 + 21.60*0.2 litros = 25.92 litros. No fim do dia 4 a quantidade de água é de (25.92 +17.175)/2 = 21.5475 (fornecimento do dia mais o que sobrou do dia anterior, menos o consumo). Como 21.5475 é maior que 20 (capacidade), foi atingido o objetivo desejado. O valor de retorno é, portanto, 4.