Universidade de Brasília
Departamento de Ciência da Computação

Introdução à Ciência da Computação - 113913

Lista de Exercícios 4

Funções Frutíferas

Observações:

- As listas de exercícios serão corrigidas por um **corretor automático**, portanto é necessário que as entradas e saídas do seu programa estejam conforme o padrão especificado em cada questão (exemplo de entrada e saída). Por exemplo, não use mensagens escritas durante o desenvolvimento do seu código como "Informe a primeira entrada". Estas mensagens não são tratadas pelo corretor, portanto a correção irá resultar em resposta errada, mesmo que seu código esteja correto.
- As questões estão em **ordem de dificuldade**. Cada lista possui 7 exercícios, sendo 1 questão fácil, 3 ou 4 médias e 2 ou 3 difíceis.
- Assim como as listas, as provas devem ser feitas na versão Python 3 ou superior.
- Leia com atenção e faça **exatamente** o que está sendo pedido.

Questão A - Função de Comparação

Escreva uma **função *compare*** que dado dois números $\boldsymbol{x}$ e $\boldsymbol{y}$, retorne 1 se x for maior que y, 0 se for igual a y, e -1 se x for menor que y. Usando o retorno da função, imprima na tela: "x e maior que y" se o retorno for 1, "x e igual a y" se o retorno for 0, e "x e menor que y", caso contrário.

Entrada

Duas linhas de entrada correspondentes aos inteiros $\boldsymbol{x}$ e $\boldsymbol{y}$.

Saída

Será impresso na tela a mensagem "x e maior que y" se o retorno da função for 1, "x e igual a y" se o retorno da função for 0, e "x e menor que y", caso contrário.

Exemplo de Entrada	**Exemplo de Saída**
4 5	x e menor que y
3 3	x e igual a y
-1 -2	x e maior que y

Tabela 1: Questão A

Questão B - Função para Entrada de Dados

Usando uma função, faça um programa que leia 10 números inteiros e imprima na tela o maior deles. No caso de valores iguais, imprima qualquer um dos maiores. Caso o maior número seja múltiplo do primeiro número $\boldsymbol{n}$ lido, imprima $\boldsymbol{n}$ na tela.

Entrada
Dez números inteiros, considere que o primeiro número lido nunca será 0.

Saída
O maior número ***maior*** e o primeiro número $\boldsymbol{n}$ lido, caso $m = a \cdot n, a \in \mathbb{Z}$.

Exemplo de Entrada	Exemplo de Saída
3 1 2 3 4 5 6 7 8 9	9 3
-1 -5 -4 10 8 0 4 3 2 1	10 -1
-2 -4 -8 -16 -32 -64 -128 -256 -512 -2	-2 -2

Tabela 2: Questão B

Questão C - Pares do Intervalo

Usando recursividade, faça um programa que dado um inteiro $\boldsymbol{n}$ positivo lido do teclado, retorne todos os números pares maiores ou iguais a dois, que são menores ou iguais a $\boldsymbol{n}$.

Entrada
Um único inteiro $\mathbf{n} \geq \mathbf{0}$.

Saída
Todos os números pares, maiores ou iguais a zero, que são menores ou iguais a $\boldsymbol{n}$, um por linha.

Exemplo de Entrada	Exemplo de Saída
10	10 8 6 4 2
15	14 12 10 8 6 4 2
4	4 2

Tabela 3: Questão C

Questão D - Soma de Sequência Par

Usando **funções recursivas**, faça um programa que dado um inteiro ***n*** lido do teclado, retorne e imprima na tela a soma de todos os números pares de 0 até ***n - 2***, incluindo ***n - 2***, se for o caso. Caso ***n*** seja menor que 0, imprima na tela "-1".

Entrada
Um único inteiro ***n***.

Saída
Será impresso na tela a soma de todos os pares de 0 até ***n - 2***. Caso ***n*** seja menor que 0 o programa deverá imprimir -1 na tela.

Exemplo de Entrada	**Exemplo de Saída**
15	42
20	90
-1	-1

Tabela 4: Questão D

Questão E - Quadrado de Pares

Usando funções faça um programa que leia um valor **n indefinidas vezes**. O programa deve encerrar quando o valor de n for zero. Para cada n lido apresente o quadrado de cada um dos valores pares (conforme formato especificado abaixo) de 1 até n, inclusive n, se for o caso.

Entrada

Inteiro $\mathbf{n} \geq \mathbf{0}$.

Saída

Será impresso na tela o quadrado de todos os números pares de 1 até n que são menores ou iguais a n, conforme exemplo abaixo.

Exemplo de Entrada	Exemplo de Saída
7 0	6ˆ2 = 36 4ˆ2 = 16 2ˆ2 = 4
1 2 0	2ˆ2 = 4
10 5 3 0	10ˆ2 = 100 8ˆ2 = 64 6ˆ2 = 36 4ˆ2 = 16 2ˆ2 = 4 4ˆ2 = 16 2ˆ2 = 4 2ˆ2 = 4

Tabela 5: Questão E

Questão F - Mínimo Múltiplo Comum

O mínimo múltiplo comum (mmc) de dois inteiros $\boldsymbol{a}$ e $\boldsymbol{b}$ é o menor inteiro positivo que é múltiplo simultaneamente de $\boldsymbol{a}$ e de $\boldsymbol{b}$. Se não existir tal inteiro positivo, por exemplo, se $\boldsymbol{a = 0}$ ou $\boldsymbol{b = 0}$, então $\boldsymbol{mmc(a,b)}$ é zero por definição. O mínimo múltiplo comum é útil em operações de soma e subtração de frações vulgares, onde é preciso um denominador comum entre as frações operadas. **Usando recursividade** faça um programa que leia dois números separados por espaço indefinidas vezes e calcule o seu mmc. O programa deve encerrar quando a entrada conter um número negativo.

Entrada

Cada linha de entrada conterá dois inteiros $\boldsymbol{a}$ e $\boldsymbol{b}$.

Saída

O mínimo múltiplo comum de $\boldsymbol{a}$ e $\boldsymbol{b}$.

Exemplo de Entrada	Exemplo de Saída
8 12 20 24 3 9 -1 0	24 120 9
4 5 2 7 13 3 -5 -5	20 14 39
4 4 0 4 7 133 4 90 0 -10	4 0 133 180

Tabela 6: Questão F

Questão G - Definição Recursiva de Strings

Seja uma **string s** definida da seguinte forma:

$$s ::= nil \mid n : s'$$

onde ***nil*** representa a string vazia, e ***n:s'*** denota a string com primeiro elemento ***n*** e cauda ***s'*** (sendo s' também uma string).
O comprimento de uma string é definido recursivamente por:

$$length(s) = \begin{cases} 0; \text{ se } s = nil \\ 1 + length(s'); \text{ se } s = a : s' \end{cases}$$

A concatenação de strings também pode ser definida por uma função recursiva:

$$concat(s1, s2) = \begin{cases} s2; \text{ se } s1 = nil \\ a : (concat(s1', s2)); \text{ se } s1 = a : s1' \end{cases}$$

O reverso de strings é definido por:

$$rev(s) = \begin{cases} s; \text{ se } s = nil \\ concat(rev(s'), (n : nil)); \text{ se } s = n : s' \end{cases}$$

Uma lista é prefixo da outra se:

$$prefix(s1, s2) = \begin{cases} True; \text{ se } s1 = nil \text{ e } s2 \neq nil \\ prefix(s1', s2'); \text{ se } s1 = a : s1' \text{ e } s2 = b : s2' \\ False; \text{ caso contrário} \end{cases}$$

Considerando o código dado abaixo e usando as definições recursivas acima, **complete** o programa abaixo.

```
def concat(s1, s2):
    if not s1:
        return s2
    else:
        return s1[0:1] + concat(s1[1:], s2)
""" Para o primeiro if vamos testar se a string s1 não é vazia.
No else, temos que s1[0:1] pega o primeiro elemento da string
e s1[1:] pega o resto (operação de fatiar) """
s1 = input()
s2 = input()
print(concat(s1, s2))
""" Função recursiva para length:
def length(s):
    if not s: (s = nil)
       return 0
    else: (s != nil)
       return 1 + length(s[1:])
Tente fazer função recursiva para mdc, fibonacci e fatorial!
"""
```

Dado duas strings ***s1*** e ***s2***, o trecho de código acima escreve na tela parte do que é especificado em **Saída**.

Entrada

A entrada consistirá apenas de duas strings ***s1*** e ***s2***. **Não terá como entrada duas strings iguais.**

Saída

Escreva na tela s1 concatenada com s2, o reverso de s1 e se s1 é prefixo de s2. **No primeiro exemplo s1 é a string vazia (nil).**

Exemplo de Entrada	Exemplo de Saída
b	b True
aaa bbb	aaabbb aaa False
cd cdd	cdcdd dc True

Tabela 7: Questão G