Universidade de Brasília
Departamento de Ciência da Computação

Introdução à Ciência da Computação - 113913

Gabarito da Lista de Exercícios 2

Funções, Condicionais e Recursividade

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, a não ser que seja pedido na questão, 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 Instâncias de Entrada serão as usadas pelo corretor e suas saídas devem estar **iguais** às apresentadas em Instâncias de Saída.

Questão 1.

```
def multiplica_string(variavel_string, entrada):
    print(variavel_string*entrada) #Usando '*' imprimimos 'texto' x vezes
#Note que a identação determina o escopo da função multiplica_string
texto = "Eric, the half a bee. "
x = int(input())
multiplica_string(texto, x)
```

Instâncias de Entrada	Instâncias de Saída
0	
4	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
5	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
6	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
7	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
8	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
9	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
10	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
11	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.
12	Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. Eric, the half a bee.

Questão 2.

```
def imprimeParidade(x):
    if(x%2 == 0): #Se o resto da divisão por 2 for 0, então o número é par
        print(x, "é par")
    else: #Caso contrário é ímpar
        print(x, "é ímpar")
    print(x+2)

n = int(input())
imprimeParidade(n)
```

Instâncias de Entrada	Instâncias de Saída
1	1 é ímpar 3
0	0 é par 2
-5	-5 é ímpar -3
7	7 é ímpar 9
9	9 é ímpar 11
-85	-85 é ímpar -83
-50	-50 é par -48
20	20 é par 22
-1	-1 é ímpar 1
-2	-2 é par 0

Questão 3.

```
def imc(h, peso_pessoa):
    indice = peso_pessoa/(h**2)
    print("%.2f"%indice)
    return indice

def classificacao(indice, peso_pessoa, h):
    if(indice < 18.5):
        print("Baixo peso")
    elif(indice <= 24.9):
        print("Peso normal")
    elif(indice <= 29.9):
        print("Sobrepeso")
        peso_necessario = 24.9*h*h
        print("%.2f"%(peso_pessoa-peso_necessario))
        #Peso mínimo necessário será aquele suficiente para atingir IMC 24.9
    elif(indice <= 34.9):
        print("Obesidade grau I")
        peso_necessario = 24.9*h*h
        print("%.2f"%(peso_pessoa-peso_necessario))
    elif(indice <= 39.9):
        print("Obesidade grau II")
        peso_necessario = 24.9*h*h
        print("%.2f"%(peso_pessoa-peso_necessario))
    else:
        print("Obesidade grau III")
        peso_necessario = 24.9*h*h
        print("%.2f"%(peso_pessoa-peso_necessario))

#Note que poderíamos fazer tudo em função só, mas usaremos duas funções
peso = float(input())
altura = float(input())
indice_pessoa = imc(altura, peso)
classificacao(indice pessoa, peso, altura)
```

Instâncias de Entrada	Instâncias de Saída
80 1.84	23.63 Peso normal
55 1	55.00 Obesidade grau III 30.10
90 1.90	24.93 Sobrepeso 0.11
100 1.90	27.70 Sobrepeso 10.11
85.54 1.84	25.27 Sobrepeso 1.24
60 1.50	26.67 Sobrepeso 3.98
62.5 1.67	22.41 Peso normal

50 1.40	25.51 Sobrepeso 1.20
50 1.75	16.33 Baixo peso
45 1.45	21.40 Peso normal

Questão 4.

```
A = float(input())
maior = A
teste = 0
B = float(input())
if(B > maior):
    maior = B
    teste = 1
C = float(input())
if(C > maior):
    maior = C
    teste = 2
""" Além de sabermos o valor da maior lado, também precisamos saber qual é.
Através da variável teste saberemos qual o maior lado """

if(teste == 0): #Nesse caso A é o maior lado
    if(maior > B + C or maior == B + C):
        print("NAO FORMA TRIANGULO")
    elif(maior*maior == B*B + C*C):
        print("TRIANGULO RETANGULO")
    elif(A == B and B == C):
        print("TRIANGULO EQUILATERO")
    elif(A == B or A == C or B == C):
        print("TRIANGULO ISOSCELES")
    else:
        print("TRIANGULO ACUTANGULO OU OBTUSANGULO")
elif(teste == 1): #Aqui B é o maior lado
    if(maior > A + C or maior == B + C):
        print("NAO FORMA TRIANGULO")
    elif(maior*maior == A*A + C*C):
        print("TRIANGULO RETANGULO")
    elif(A == B and B == C):
        print("TRIANGULO EQUILATERO")
    elif(A == B or A == C or B == C):
        print("TRIANGULO ISOSCELES")
    else:
        print("TRIANGULO ACUTANGULO OU OBTUSANGULO")
else: #Caso contrário C será o maior
    if(maior > A + B or maior == B + C):
        print("NAO FORMA TRIANGULO")
    elif(maior*maior == B*B + A*A):
        print("TRIANGULO RETANGULO")
    elif(A == B and B == C):
        print("TRIANGULO EQUILATERO")
    elif(A == B or A == C or B == C):
        print("TRIANGULO ISOSCELES")
    else:
        print("TRIANGULO ACUTANGULO OU OBTUSANGULO")
""" Podemos usar elif, visto que não é possivel entrar com raiz quadrada
no Python Shell, seja x o valor de dois lados do triângulo e y do terceiro.
Para que y^2 seja igual a 2*(x^2), é necessário que y seja igual a
(2^(1/2)) * x"""
```

Instâncias de Entrada	Instâncias de Saída
7.0 7.0 7.0	TRIANGULO EQUILATERO
3.0 4.0 5.0	TRIANGULO RETANGULO
3 2 1	NAO FORMA TRIANGULO
1.50 1.45 1.30	TRIANGULO ACUTANGULO OU OBTUSANGULO
3.0 3.5 4.95	TRIANGULO ACUTANGULO OU OBTUSANGULO
3.0 3.5 4.25	TRIANGULO ACUTANGULO OU OBTUSANGULO
7.5 3 4.5	NAO FORMA TRIANGULO
4 4 4	TRIANGULO EQUILATERO
5 5 4	TRIANGULO ISOSCELES
3.0 3.0 1.5	TRIANGULO ISOSCELES

Questão 5.

```
hi, mi, hf, mf = input().split()
hi, mi, hf, mf = [int(hi), int(mi), int(hf), int(mf)]

""" Vamos resolver esse problema por casos.
Note que não é possivel ter um caso em que hf == hi e mf > mi, pois o jogo
tem duração máxima de 24 horas.
"""
if(hf <= hi and mf < mi): #Exemplo: 7 5 6 4 - 22h59m ou 7 5 7 4 - 23h59m
    """Vamos transformar as horas para minutos e retirarmos os minutos
    já calculados assim é possível ter as horas certas nos dois exemplos acima,
    fazendo a divisão inteira por 60 """
    minutos = (60-mi)+mf
    horas = ((24 - hi)+hf) * 60 - minutos
    print("O jogo durou %d hora(s) e %d minuto(s)."%(horas//60, minutos))
elif(hf == hi and mf == mi):
    print("O jogo durou 24 hora(s) e 0 minuto(s).")
elif(hf < hi and mf >= mi): #Exemplo: 7 5 6 5 - 23h0m ou 7 5 6 6 - 23h1m
    horas = (24 - hi) + hf
    print("O jogo durou %d hora(s) e %d minuto(s)."%(horas, mf-mi))
else: # Exemplo: 7 5 8 4 - 0h59m  ou 7 8 9 10 - 2h2m
    """Vamos seguir o mesmo raciocínio para o primeiro caso, aqui teremos
    os casos em que: hi < hf and mi < mf(simples) ou hi < hf and mi >= mf """
    minutoi = hi * 60 + mi
    minutof = hf * 60 + mf
    horas = (minutof - minutoi)//60
    minutos = (minutof - minutoi)%60
    print("O jogo durou %d hora(s) e %d minuto(s)."%(horas, minutos))
```

Instâncias de Entrada	Instâncias de Saída
6 5 6 4	O jogo durou 23 hora(s) e 59 minuto(s).
7 5 6 4	O jogo durou 22 hora(s) e 59 minuto(s).
8 8 8 8	O jogo durou 24 hora(s) e 0 minuto(s).
8 10 8 5	O jogo durou 23 hora(s) e 55 minuto(s).
8 5 9 3	O jogo durou 0 hora(s) e 58 minuto(s).
6 4 7 1	O jogo durou 0 hora(s) e 57 minuto(s).
7 5 6 5	O jogo durou 23 hora(s) e 0 minuto(s).
7 5 6 6	O jogo durou 23 hora(s) e 1 minuto(s).
7 5 8 4	O jogo durou 0 hora(s) e 59 minuto(s).
7 8 9 10	O jogo durou 2 hora(s) e 2 minuto(s).

Questão 6.

```
def fatorial(num):
    if (num == 1 or num == 0):
        return 1
    else:
        return num*fatorial(num-1)
""" Após fazer as subtrações, quando num chegar a 1 então terminaremos as
chamadas, pois não há necessidade de fazer fatorial de 0. Considerando, é claro,
que o num seja maior que 1, caso seja 0 ou 1 então já retornaremos 1.
Enquanto num for maior que 1, precisamos fazer num*fatorial(num-1) e após
finalizar todos as chamadas de funções encadeadas teremos o fatorial de num."""

num1 = int(input())
num2 = int(input())
print("%d! + %d! ="%(num1,num2), fatorial(num1) + fatorial(num2))
```

Instâncias de Entrada	Instâncias de Saída
0 0	0! + 0! = 2
1 5	1! + 5! = 121
4 7	4! + 7! = 5064
0 5	0! + 5! = 121
10 10	10! + 10! = 7257600
7 6	7! + 6! = 5760
1 1	1! + 1! = 2
1 0	1! + 0! = 2
3 3	3! + 3! = 12
7 13	7! + 13! = 6227025840

Questão 7.

```
def Fibonacci(n):
    if(n == 1 or n == 2):
        return 1
    else:
        return Fibonacci(n-1) + Fibonacci(n-2)
""" Para a função recursiva Fibonacci foi aplicada a definição, da mesma
forma para o cálculo do fatorial vamos fazendo chamadas de funções dentro da
chamada da função e retornando até que n chegue a 1 ou 2 """

n = int(input())
quantidade_casais = Fibonacci(n)
if(quantidade_casais % 2 == 0): #Nesse caso a quantidade é par
    print(quantidade_casais)
    print(Fibonacci(n-1))
else:
    print(quantidade_casais)
""" Note que a quantidade de casais que irão nascer no próximo mês, pela
definição de Fibonacci é justamente a quantidade de casais do mês anterior,
ou seja, Fibonacci(n-1). Pois Fibonacci(n+1) = Fibonacci(n) + Fibonacci(n-1),
assim Fibonacci(n+1) - Fibonacci(n) = Fibonacci(n-1) """
```

Instâncias de Entrada	Instâncias de Saída
2	1
4	3
11	89
12	144 89
13	233
8	21
9	34 21
17	1597
3	2 1
5	5