****

**Introdução à Ciência da Computação - 113913**

**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 questões estão em **ordem de dificuldade**. Cada lista possui 7 exercícios, sendo 1 questão fácil, 3 ou 4 médias e 2 ou 3 difíceis.
* Leia com atenção e faça **exatamente** o que está sendo pedido.

1. “Eric, the half a bee” é uma música do grupo humorístico britânico Monty Python. A linguagem Python foi batizada em homenagem ao grupo. Faça uma função que receba como parâmetro a string***s*** “Eric, the half a bee. ” e um valor inteiro positivo ***x***, lido do teclado. A seguir imprima a string ***s*** na tela ***x*** vezes.

**Entrada**

Um único número natural ***x***, ***x ≥ 0***.

**Saída**

Imprima na tela a string ***x*** vezes. Haverá um espaço após o último ponto final.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 2 | Eric, the half a bee. Eric, the half a bee. |
| 1 | Eric, the half a bee. |
| 3 | Eric, the half a bee. Eric, the half a bee. Eric, the half a bee. |

1. Faça um programa que use uma função para imprimir na tela se um número lido do teclado é par ou ímpar. Se for par imprima também o próximo número par, caso contrário imprima o próximo ímpar.

**Entrada**

Número inteiro ***n***.

**Saída**

A saída conterá duas linhas, uma informando se o número é par ou ímpar e outra mostrando o próximo par ou ímpar, conforme exemplo abaixo.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 2 | 2 é par  4 |
| 3 | 3 é ímpar  5 |
| -4 | -4 é par  -2 |

1. Usando como dados de entrada altura ***h*** (em metros) e peso ***p*** (em quilos), elabore um programa que calcule o IMC (índice de massa corporal) do usuário, usando a fórmula:

Depois interprete e informe o resultado, da seguinte forma:

* **Baixo peso:** IMC abaixo de 18,5
* **Peso normal:** IMC entre 18,5 e 24,9
* **Sobrepeso:** IMC entre 24,9 e 29,9
* **Obesidade grau I:** IMC entre 29,9 e 34,9
* **Obesidade grau II:** IMC entre 34,9 e 39,9
* **Obesidade grau III:** IMC maior que 39,9

Caso ele esteja acima da faixa de peso normal, informe também o mínimo de quilos que serão necessários perder (mantendo a altura, é claro) para chegar à faixa peso normal. Use **função** para fazer o cálculo do IMC.

**Entrada**

Duas linhas de dados: peso (em kg) e altura (em metros).

**Saída**

Caso o usuário esteja na faixa Baixo peso ou Peso normal apenas imprima a mensagem informando o IMC e, na próxima linha, a sua classificação correspondente usando a tabela, conforme exemplo fornecido abaixo. Caso contrário, imprima também o peso **mínimo** necessário a se perder para chegar à faixa peso normal com 2 casas decimais após a vírgula.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 79  1.84 | 23.33  Peso normal |
| 84.4  1.85 | 24.66  Peso normal |
| 85  1.60 | 33.20  Obesidade grau I  21.26 |

1. Leia 3 valores de ponto flutuante A, B e C e ordene-os de modo que A representa o maior dos 3 lados. A seguir, determine o tipo de triângulo que estes três lados formam, com base nos seguintes casos, sempre escrevendo uma mensagem adequada:

* Se , apresente a mensagem: **NAO FORMA TRIANGULO**
* Se , apresente a mensagem: **TRIANGULO RETANGULO**
* Se os três lados forem iguais, apresente a mensagem: **TRIANGULO EQUILATERO**
* Se apenas dois dos lados forem iguais, apresente a mensagem: **TRIANGULO ISOSCELES**
* Caso contrário, apresente a mensagem: **TRIANGULO ACUTANGULO OU OBTUSANGULO**

**Entrada**

A entrada contém 3 valores reais todos maiores que zero. Não terá como entrada um valor tal que o triângulo seja retângulo e isósceles ao mesmo tempo.

**Saída**

Imprima a classificação do triângulo.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 7.0  5.0  7.0 | TRIANGULO ISOSCELES |
| 6.0  6.0  6.0 | TRIANGULO EQUILATERO |
| 1.0  3.0  1.0 | NAO FORMA TRIANGULO |

1. Leia a hora inicial, minuto inicial, hora final e minuto final de um jogo. A seguir calcule a duração do jogo, considerando que o jogo pode acabar em um dia e terminar em outro, tendo uma duração máxima de 24 horas.

**Entrada**

Quatro números inteiros representando a hora de início e fim do jogo.

**Saída**

Mostre a seguinte mensagem: “O jogo durou XX hora(s) e YY minuto(s).”

**Nota:** Lembre-se que para ler vários valores em uma mesma linha, use ***input().split()***. Se o argumento de split for vazio, o separador das variáveis é um espaço em branco. Porém, lembre-se que input lê apenas strings do teclado, portanto você deverá converter as strings em inteiros. No exemplo a seguir, o usuário digita valores separados por um espaço em branco e aperta enter para enviá-los, então, o programa lê esses valores separados por espaços como strings (na ordem em que aparecem), guardados nas variáveis correspondentes e os converte para int:

***A, B, C, D = input().split()***

***A, B, C, D = [int(A), int(B), int(C), int(D)]***

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 7 5 7 4 | O jogo durou 23 hora(s) e 59 minuto(s). |
| 7 7 7 7 | O jogo durou 24 hora(s) e 0 minuto(s). |
| 7 10 8 9 | O jogo durou 0 hora(s) e 59 minuto(s). |

1. Usando recursividade faça um programa que lê do teclado dois números inteiros

***num1*** e ***num2*** e imprima na tela ***num1!*** + ***num2!*** (soma dos fatoriais).

**Entrada**

Dois inteiros ***num1 ≥ 0*** e ***num2 ≥ 0***.

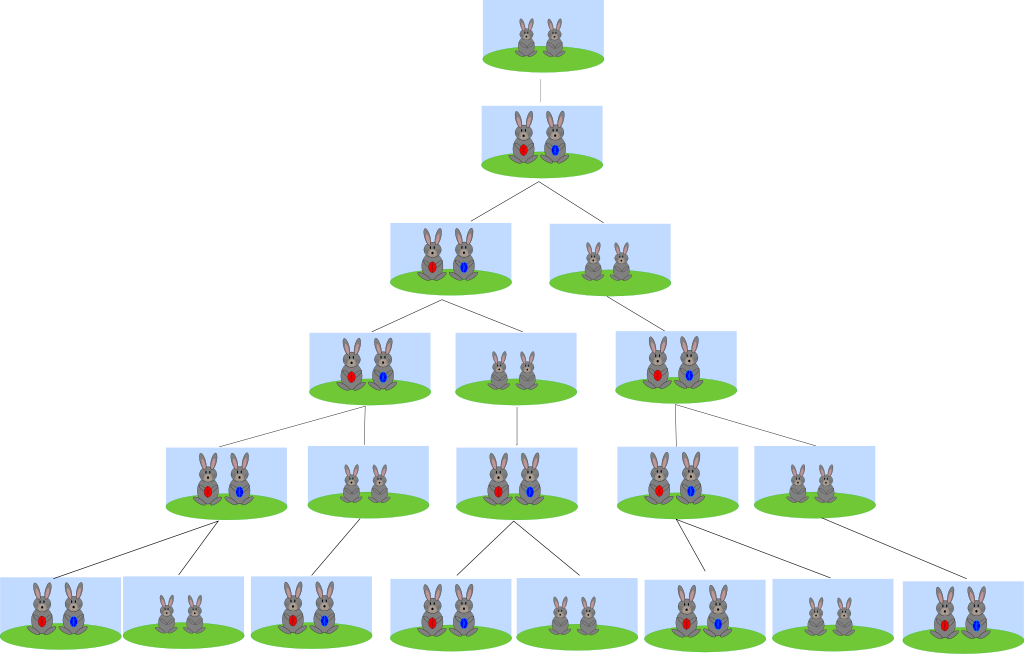
**Saída**

O programa imprimirá na tela a mensagem: “***num1!*** + ***num2!*** = ***num1!*** + ***num2!***”, conforme exemplo abaixo.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 4  4 | 4! + 4! = 48 |
| 0  1 | 0! + 1! = 2 |
| 15  10 | 15! + 10! = 1307677996800 |

1. No ocidente, a sequência de Fibonacci apareceu pela primeira vez no *livro Liber Abaci* (1202) de Leonardo Fibonacci embora ela já tivesse sido descrita por gregos e indianos. Fibonacci considerou o crescimento de uma população idealizada (não realista biologicamente) de coelhos. Os números descrevem o número de casais na população de coelhos depois de ***n*** meses se for suposto que:

* No primeiro mês nasce apenas um casal,
* casais amadurecem sexualmente (e reproduzem-se) apenas após o segundo mês de vida,
* não há problemas genéticos no cruzamento consaguíneo,
* todos os meses, cada casal fértil dá a luz a um novo casal, e
* os coelhos nunca morrem.

***Ilustração representativa da série de Fibonacci, demonstrando o crescimento populacional de coelhos.***

Sendo a quantidade de casais após ***n*** meses, usando recursividade faça um programa que, dado um inteiro positivo ***n*** digitado pelo usuário, calcule o n-ésimo termo da sequência de Fibonacci usando a definição dada abaixo:

**-**

**-**

Caso haja um número par de casais de coelhos após ***n*** meses, imprima também quantos novos casais de coelhos vão nascer no próximo mês.

**Entrada**

Inteiro ***n > 0***, onde ***n*** representa os meses que passaram.

**Saída**

Será impresso na tela o número de casais após ***n*** meses, e caso esse número seja par será impresso também quantos novos casais irão nascer no próximo mês.

|  |  |
| --- | --- |
| **Exemplo de Entrada** | **Exemplo de Saída** |
| 6 | 8  5 |
| 1 | 1 |
| 10 | 55 |