Trabalho 1

LEIA ATENTAMENTE AS REGRAS E OS ENUNCIADOS

R E G R A S

- O trabalho deverá ser realizado individualmente.
- O trabalho deverá ser enviado para o Microsoft Teams até o dia 27/04/2023 (quinta-feira).
- A data de entrega não será adiada.
- Os 3 programas solicitados (arquivos .PY) deverão ser compactados em um único arquivo (apenas .ZIP) com o nome e sobrenome do aluno, seguido da matrícula, conforme o exemplo abaixo.
  - Exemplo: **JoseBonifacioDeAndrade_20212210999.zip**
- Não envie outros arquivos dentro do ZIP. Somente os arquivos com extensão PY.
- Os programas (arquivos .PY) deverão ter os nomes conforme definido nos enunciados.
- Não serão aceitos trabalhos enviados por e-mail.
- Trabalhos com estruturas e/ou organizações semelhantes (plágio) serão penalizados com a nota zero.
- Cada arquivo PY deve ser passível de compilação e de posterior execução sem erros.
- O programa que não obedecer às restrições estabelecidas receberá zero.
- A interpretação do enunciado de cada questão faz parte da resolução da questão.
- Este trabalho possui nota total igual a 10,0 (dez pontos).
- Este trabalho corresponde a 15% da Nota Final da disciplina.
- Antes de escrever o código, faça o estudo do problema e o planejamento da sua solução.
- Lembre-se de documentar seu código.

1) Programa: triangular.py (2,0 pontos)

Um número inteiro **n** é chamado de **triangular par** se ele é resultado da multiplicação de três números pares consecutivos $\mathbf{x_1}$, $\mathbf{x_2}$ e $\mathbf{x_3}$ ($x_1 > 0$, $x_2 > 0$ e $x_3 > 0$). Exemplo: 48 = 2 x 4 x 6. Por outro lado, um número inteiro **n** é chamado de **triangular ímpar** se ele é resultado da multiplicação de três números ímpares consecutivos $\mathbf{x_1}$, $\mathbf{x_2}$ e $\mathbf{x_3}$ ($x_1 > 0$, $x_2 > 0$ e $x_3 > 0$). Exemplo: 105 = 3 x 5 x 7.

Crie um programa em Python para ler um valor inteiro **k**, o **tipo** (P-triangular par ou I-triangular ímpar) e imprimir os **k primeiros** números triangulares pares ou ímpares, conforme solicitado pelo usuário.

Dica: use o tipo **long** para as variáveis inteiras.

Restrições:

a) $k > 0$. Se $k \leq 0$, então solicite novamente o valor até o usuário digitar corretamente.

b) Tipo = P ou I. Se tipo ≠ P e I, então solicite novamente o tipo até o usuário digitar corretamente.

c) Não poderão ser usados vetores nem qualquer outro tipo de estrutura de dados, somente variáveis simples.

d) Todo o código deverá estar implementado em uma única, sem o uso de funções auxiliares.

2) Programa: serie.py (2,0 pontos)

O valor de H é calculado pela seguinte série:

$$\mathrm{H} = \frac{1}{n} - \frac{2}{(n-1)^{2!}} + \frac{3}{(n-2)^{3!}} - \frac{4}{(n-3)^{4!}} + \frac{5}{(n-4)^{5!}} + \cdots + \frac{n}{1}$$

Crie um programa em Python que lê um valor **n** (n > 0) onde **n** é o número de termos da série e imprime o valor de H, calculado de acordo com a fórmula acima. Deverá existir uma função **cal_fat.py** exclusiva para o cálculo do fatorial.

Restrições:

a) n > 0. Se n ≤ 0, então solicite novamente o valor até o usuário digitar corretamente.

b) Não poderá ser usada nenhuma função matemática implementada na biblioteca do Python nem em nenhuma outra biblioteca.

c) Não poderão ser usados vetores nem qualquer outro tipo de estrutura de dados, somente variáveis simples.

d) Todo o código deverá estar implementado na função **main** e da sub-rotina **cal_fat**, sem o uso de outras funções auxiliares, com a exceção das funções **input** e **print**, e da função **pow** contida na biblioteca matemática **math**.

3) Programa: orçamento.py (3,0 pontos)

Uma empresa de reformas de casas e apartamentos deseja criar um programa para calcular o orçamento de instalação de ar-condicionado. O valor do orçamento é calculado levando-se em conta a mão de obra, a quantidade de serviço a ser executado, o tipo do acabamento desejado pelo cliente e o desconto.

$$valor\ do\ orçamento = (mão\ de\ obra + obra\ bruta)$$

Para o cálculo da mão de obra, além do técnico em ar-condicionado, a empresa trabalha com outros 3 tipos de profissionais: pedreiro, eletricista e pintor. Todo profissional sempre trabalha com um ajudante. O valor da diária de cada profissional e do ajudante é dado pela tabela a seguir. É importante notar que uma reforma pode envolver um ou mais profissionais trabalhando por diferentes horários. Por exemplo: uma reforma pode precisar, além de 10 horas do técnico em ar-condicionado, também de um pedreiro trabalhando por 15 horas, um eletricista trabalhando por 3 horas e um pintor trabalhando por 2 horas.

Profissional	Diária
Pedreiro	R$ 11,00
Eletricista	R$ 13,00
Técnico em Ar-Condicionado	R$ 15,00
Pintor	R$ 12,00
Ajudante	R$ 5,00

O valor da obra bruta é calculado com base na quantidade de metros cúbicos do ambiente em que o ar-condicionado será instalado. Se a instalação for realizada no cômodo em uma casa (indicado pelo caractere ‘**c**’ ou ‘**C**’) serão cobrados R$ 40,00/metro^2, porém se a reforma for em um apartamento (indicado pelo caractere ‘**a**’ ou ‘**A**’) serão cobrados R$ 50,00/metro^2.

Crie um programa em Python para ler **todos** os dados necessários para calcular o orçamento de uma reforma, conforme detalhado anteriormente, e imprimir:

- Valor em R$ da mão de obra por profissional (valor do pedreiro, do eletricista, etc.);
- Valor em R$ da obra bruta;
- Valor em R$ do orçamento;

O programa deve conter e fazer uso de uma função nomeada **calculaMaodeObra** que retorna o valor total da mão de obra. Como parâmetros de entrada, as horas de cada profissional.

O programa deve conter e fazer uso de uma função nomeada **calculaTipoResidencia**, que retorna o valor da obra bruta, e que deve ter como parâmetros de entrada o tipo de residência e a área em m² do cômodo.

Além dessas funções, o usuário deve ser informado do valor da mão de obra de cada profissional separadamente e o valor da obra.

Restrições:

a) Todo o código deverá estar implementado ou rotina **main**, e nas sub-rotinas **calculaMaodeObra** e **calculaTipoResidencia**, sem a criação ou uso de quaisquer outras rotinas auxiliares, com a exceção das funções **input** e **print**.

b) As sub-rotinas **calculaMaodeObra** e **calculaTipoResidencia** não devem ter qualquer interação com o usuário do programa.

c) Não poderão ser usados vetores nem qualquer outro tipo de estrutura de dados, somente variáveis simples.

4) Programa: bissecao.py (3,0 pontos)

Na Engenharia, existe a necessidade de se determinar a raiz de uma função, ou seja, $f(x) = 0$. O valor x encontrado é chamado de raiz da equação ou zero da função. Para equações de 1º e 2º graus, as raízes exatas podem ser encontradas por métodos analíticos (como a fórmula de Bhaskara). Com o aumento do grau da equação, encontrar exatamente a raiz da equação pode não ser tão obvio ou pode não ter um método analítico. Para isso, existem métodos que, dado a precisão que se deseja encontrar a raiz, esta é encontrada numericamente.

O método da bisseção é o método numérico de busca de raízes mais simples na literatura. Ele funciona da seguinte forma:

- Dada uma função contínua dentro de um intervalo ($f: [a, b] \to \mathbb{R}, u = f(x)$, tendo $f(a)$ e $f(b)$ sinais opostos, ou seja, $f(a) \cdot f(b) < 0$. Com essas condições, o teorema do valor intermediário garante a existência de uma raiz no intervalo (a, b).
- Dividindo o intervalo no seu ponto médio $c = \frac{a+b}{2}$, verificar em qual dos dois subintervalos garante-se a existência de uma raiz. Para tanto, basta verificar se $f(a) \cdot f(c) < 0$. Caso afirmativo, existe pelo menos uma raiz no intervalo (a, c), caso contrário garante-se a existência de uma raiz no intervalo (c, b). O procedimento é, então, repetido para o subintervalo correspondente à raiz até que c aproxime a raiz com a precisão desejada, ou seja, $f(x) < precisao$.
- A partir do intervalo inicial dado e da precisão, pode-se encontrar o número de iterações máximas necessárias para a aproximação requerida. Essa quantidade máxima é dada pela seguinte fórmula:

$$n = \frac{\log(a - b) - \log(precisao)}{\log 2}$$

Implemente um programa Python que <u>leia</u> todos os dados necessários para calcular a raiz de uma função pelo método de bisseção e <u>imprimir</u> a quantidade de iterações máxima necessária para encontrar a raiz e a própria raiz.

Teste seu programa para as seguintes funções:

i. $f(x) = x^3 - 6x^2 - x + 30$

ii. $f(x) = x + \log x$

iii. $f(x) = 3x - \cos(x)$

iv. $f(x) = x^3 - e^{2x} + 3$

v. $f(x) = \sin(x) - \ln x$

Restrições:

a) Todo o código deverá estar implementado ou rotina **main**, sem a criação ou uso de quaisquer outras rotinas auxiliares, com a exceção das funções **input**, **print** e funções da biblioteca **math**.

b) Não poderão ser usados vetores nem qualquer outro tipo de estrutura de dados, somente variáveis simples.

Dicas:

- Tente fazer na mão uma das equações acima para ter a certeza de que entendeu o método.
- Pesquise a biblioteca **math** para funções de logaritmos, potenciação e trigonométricas.