Lista de exercícios semana 1

1. **(1 ponto)** Desenvolva um programa para calcular quanto uma pessoa deve pagar de imposto de renda, considerando que ele é calculado a partir da multiplicação do salário pela taxa correspondente.

Renda	Imposto de Renda
de 0.00 a R$ 2000.00	Isento
de R$ 2000.01 até R$ 3000.00	8 %
de R$ 3000.01 até R$ 4500.00	18 %
acima de R$ 4500.00	28 %

O valor deve ser impresso com duas casas decimais.

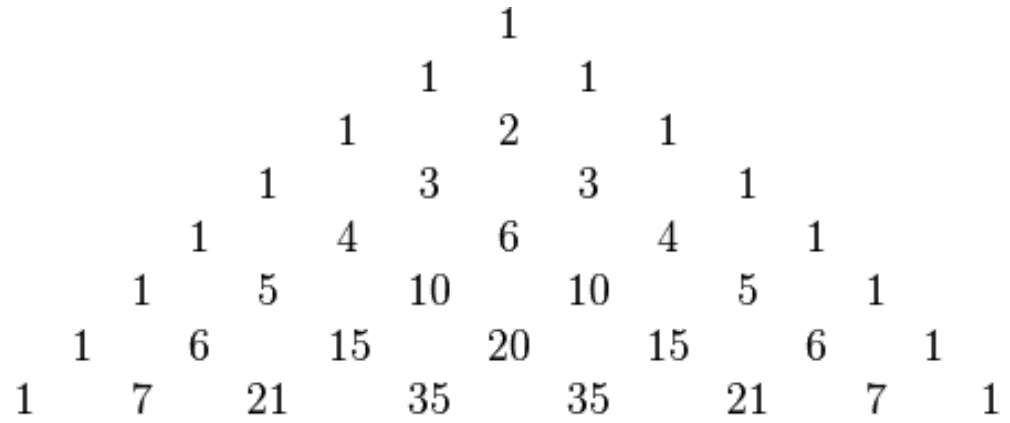
Entrada: 3002.00
Saída: R$ 540.36
Entrada 2: 1701.12
Saída 2: Isento
Entrada 3: 4520.00
Saída 3: R$ 1265.60

2. **(1,5 pontos)** Escreva uma função que dado um inteiro positivo **n**, imprima o número primo mais próximo menor ou igual a **n**. Se a entrada for menor que dois (n < 2), a função deve retornar "Não foi possível encontrar um número primo".
DICA: pode ser interessante construir uma função auxiliar que recebe um número como parâmetro e determina se este número é primo ou não.

Entrada: 46
Saída: 43
Entrada 2: 10
Saída 2: 7
Entrada 3: 15
Saída 3: 13
Entrada 4: 0
Saída 3: Não foi possível encontrar um número primo

3. **(1,5 pontos)** O triângulo de Pascal é um triângulo numérico infinito formado por números binomiais. Cada número do triângulo de Pascal é igual

à soma do número imediatamente acima e do antecessor do número de cima.



A soma dos valores pertencentes à i-ésima linha de um triângulo de pascal é 2^i, sendo i=0 na primeira linha ($2^0 = 1$), i=1 na segunda ($2^1 = 2 = 1 + 1$) e assim por diante. Escreva uma função que recebe um inteiro **n** e retorna a soma de **todos** os valores de um triângulo de pascal que possui **n** linhas.

Entrada: 1
Saída: 1
Entrada: 2
Saída: 3
Entrada: 5
Saída: 31

4. **(2 pontos)** Escreva um programa que, dada uma lista de números [-2, 34, 5, 10, 5, 4, 32] qualquer, retorne um array de hashes com as seguintes informações: o menor valor e sua posição, o maior valor e sua posição, e a mediana e sua posição caso a lista tenha tamanho ímpar. Em seguida, imprima a lista ordenada. *** Obs. Para listas com tamanho ímpar, a mediana é o valor do meio, quando ordenada (.sort). Para listas com tamanho par, a mediana é o valor da média aritmética entre os dois números do meio. A posição deve ser indicada por um array com a posição dos dois números.

Entrada: [1, 2 , 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Saída: [{:menor=>1, :posicao=>0}, {:maior=>15, :posicao=>14}, {:mediana=>8, :posicao=>7}]
ordenada: [1, 2 , 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
Entrada 2: [-10, -15, -20, 10, 20, 5]

Saída 2: [{:menor=>-20, :posicao=>2}, {:maior=>20, :posicao=>4}, {:mediana=>-2.5, :posicao=>[2, 3]}]
ordenada: [-20, -15, -10, 5, 10, 20]}

5. **(2 pontos)** Um restaurante tem seu cardápio organizado em um hash. O hash consiste em três chaves, “entrada”, “principal” e “sobremesa”. Você deverá colocar de 3 a 5 pratos vinculados a cada uma dessas chaves (array de pratos).

{:entrada=>["Bruschetta", "Salada","Sopa"], :principal=> ...}

Escreva um programa que retorne um array com um prato selecionado aleatoriamente de cada chave e os imprima na tela.

Coloquei acima o começo de como ficará uma possível versão do seu hash. Visto que temos um hash com as chaves entrada, principal e sobremesa. O resultado esperado no array gerado será um prato de cada uma dessas chaves, selecionado aleatoriamente.

Exemplo de resultado: ["Bruschetta", "Bobó de Camarão", "Torta de Limão"]

6. **(2 pontos)** Escreva uma função que receba um número (de 1 a 10) e retorne dois arrays: o primeiro contendo a sequência de números de 1 até o número fornecido à função, e o segundo contendo os fatoriais dos números ímpares dessa lista.

Entrada: 6
Saída: [1, 2, 3, 4, 5, 6]
[1, 6, 120]

Entrada: 9
Saída: [1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 6, 120, 5040, 362880]

DESAFIO

(1 ponto extra) Dada uma matriz (array de arrays de mesmo tamanho), faça um programa que remova a coluna do meio da matriz, e retorne a matriz

transposta. Caso a matriz tenha um número par de colunas, remova as duas centrais.

Dica: Confira a documentação do ruby, isso é mais fácil do que parece

Entrada 1:

[[1, 2, 3],
[4, 5, 6],
[7, 8, 9],
[0, 1, 2]]

Saída 1:

[[1, 4, 7, 0],
[3, 6, 9, 2]]

Entrada 2:

[[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]

Saída 2:

[[1, 4, 7],
[3, 6, 9]]

Obs.: Nesses exemplos quebramos as linhas das entradas e saídas dos arrays para ficar mais claro o funcionamento do programa, mas não é necessário colocar as entradas e saídas formatadas dessa forma.