MC102 - Algoritmos e Programação de Computadores

Lista de Exercícios 5

1. Considere o código em Python abaixo:

```
j = 1

def main():
  a = 9

  if a % 2 == 0:
    a = 2
  else:
    a = 3

  print(fun1(2,4))

  for i in range(3):
    for j in range(3):
      print(fun1(a, i+j))

def fun1(a, b):
  p = 1
  for i in range(b):
    p = p * a
  return p+j

main()
```

   (a) Determine quais são as variáveis locais e globais deste programa. Para cada variável local identifique a que função ela pertence.

   (b) Mostre o que será impresso na tela do computador quando for executado este programa.

2. Escreva uma função que receba dois números inteiros positivos a e b como parâmetro e determine se eles são amigos ou não, devolvendo `True` caso sejam amigos e `False` caso contrário.

   Dois números são amigos se cada número é igual à soma dos divisores próprios do outro (os divisores próprios de um número m são os divisores estritamente menores que m). Por exemplo, os divisores próprios de 220 são 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 e 110, cuja soma é 284; e os divisores próprios de 284 são 1, 2, 4, 71 e 142, cuja soma é 220. Logo, 220 e 284 são números amigos.

   A seguinte função deve ser implementada:

```
def amigos(a, b):
```

3. Escreva uma função que receba um valor inteiro positivo n como parâmetro e devolva o menor valor inteiro b, tal que $b^k = n$ para algum inteiro k. Por exemplo, se $n = 27$ então o valor devolvido deve ser 3. Já se $n = 12$, o valor devolvido deve ser 12.

   A seguinte função deve ser implementada:

```
def menor_base_log(n):
```

4. Um inteiro positivo n é **pitagórico** se existem inteiros positivos a e b tais que $a^2 + b^2 = n$. Por exemplo, 13 é pitagórico, pois $2^2 + 3^2 = 13$.

   (a) Escreva uma função que receba como parâmetro três inteiros a, b e n, e devolva `True` caso $a^2 + b^2 = n$ e `False`, caso contrário.

   A seguinte função deve ser implementada:

```
def teste(a, b, n):
```

   (b) Utilize a função do item anterior e escreva uma outra função que receba como parâmetro um inteiro positivo n e verifique se n é pitagórico, devolvendo `True` caso n seja pitagórico e `False`, caso contrário.

   A seguinte função deve ser implementada:

```
def pitagorico(n):
```

5. Escreva uma função que receba uma lista de números reais e devolva a média aritmética dos números da lista.

   A seguinte função deve ser implementada:

```
def media(v):
```

6. Escreva uma função que receba uma lista de números reais e devolva o desvio padrão dos números da lista usando a seguinte fórmula:

$$\sqrt{\frac{1}{n-1}(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2)}$$

   onde n é o número de elementos.

   A seguinte função deve ser implementada:

```
def desvioPadrao(v):
```

7. Escreva uma função chamada `sanduiche_primo` que receba como parâmetro um inteiro n e devolva uma tupla com dois valores inteiros $p1$ e $p2$, onde $p1$ é o maior número primo que é menor do que n e $p2$ é o menor número primo que é maior do que n.

   A seguinte função deve ser implementada:

```
def sanduiche_primo(n)
```

8. Escreva uma função que receba como parâmetro uma lista de inteiros. A função deve devolver uma tupla com dois valores inteiros $f1$ e $f2$, onde $f1$ é o elemento da lista com menor frequência (menor número de ocorrências na lista) e $f2$ é o elemento com maior frequência. Dica: use um dicionário para computar as frequências dos elementos da lista.

   A seguinte função deve ser implementada:

```
def frequencias(v)
```

9. Suponha que uma matriz binária `mat` represente ligações entre cidades, onde, se uma posição `mat[i][j]` possui o valor 1, então há uma estrada da cidade `i` para a cidade `j`. Seja o seguinte exemplo de matriz:

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Neste caso, há estradas da cidade 0 para as cidades 1 e 2, e da cidade 2 para a cidade 0. Para cada item abaixo escreva uma função `verifica(mat)` que receba como parâmetro uma matriz quadrada `mat`, indicando as estradas entre as cidades, e devolva uma lista `resposta`.

- Escreva uma função para determinar as cidades com estradas chegando, mas sem estradas saindo da cidade, indicando isto na lista `resposta`, tal que `resposta[i]` recebe `True` caso a cidade `i` satisfaça esta propriedade e `False`, caso contrário.
- Escreva uma função para determinar as cidades com estradas saindo, mas sem estradas chegando na cidade, indicando isto na lista `resposta`, tal que `resposta[i]` recebe `True` caso a cidade `i` satisfaça esta propriedade e `False`, caso contrário.
- Escreva uma função para determinar as cidades isoladas (sem estradas chegando ou saindo da cidade), indicando isto na lista `resposta`, tal que `resposta[i]` recebe `True` caso a cidade `i` satisfaça esta propriedade e `False`, caso contrário.

10. No jogo Sudoku temos uma matriz 9×9 dividida em 9 quadrados de 3×3 preenchidos previamente com alguns números entre 1 e 9 (veja o exemplo à esquerda abaixo). Uma solução para uma instância do jogo consiste no preenchimento de todas as posições vazias com números entre 1 e 9 respeitando-se as seguintes regras:

(a) Não pode haver números repetidos em um mesmo quadrado, ou seja, cada número entre 1 e 9 deve aparecer exatamente uma vez em cada quadrado.

(b) Não pode haver números repetidos em nenhuma linha da matriz.

(c) Não pode haver números repetidos em nenhuma coluna da matriz.

Escreva uma função que receba uma matriz 9×9 como parâmetro, que represente uma proposta de solução para um Sudoku, e teste se a matriz é uma solução válida para o jogo, devolvendo `True` em caso verdadeiro e `False`, caso contrário.

A seguinte função deve ser implementada:

```
def solucao(mat):
```

Veja abaixo um exemplo (à direita) de uma solução para um Sudoku.

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Sudoku não resolvido

4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Solução