Computação I - Modelo de Prova

A prova é individual e sem consulta. Responda as questões na folha de respostas, a lápis ou a caneta. Se tiver qualquer dúvida consulte o professor. A prova é frente e verso!

Boas práticas de programação também são parte da avaliação. Lembre-se de usar nomes significativos para variáveis e funções, organizar seu código com funções, colocar comentários e identar corretamente.

1. (2 pontos) Para cada item abaixo, escreva uma função que:
   (a) recebe um número inteiro n e uma lista de inteiros L e retorna a soma de todos os valores de L que não sejam múltiplos de n.
   (b) recebe dois inteiros e retorna o mmc (menor múltiplo comum) deles. Para isto, siga a seguinte ideia: sendo m e n inteiros com $m \leq n$ multiplique m sucessivamente por 1,2,3,... até que o primeiro destes produtos seja múltiplo de n, este produto é o mmc.

2. (2 pontos) Seja a seguinte função:

```
# Processa cada palavra de uma frase.
# str -> int
def processa(frase):
    # Separa as palavras da frase numa lista.
    palavras = str.split(frase)
    s = 0
    # Analisa cada palavra.
    for x in palavras:
        if s>6 and s<=8:
            break
        if len(x)%3==0:
            continue
        if 'e' in x:
            s += 1
        else:
            s += len(x)
    return s
```

   (a) Reescreva esta função usando *while* no lugar de for.
   (b) Informe qual é o valor retornado pela função nas seguintes chamadas:
      (b.1) $processa('I\ am\ not\ in\ danger,\ I\ am\ the\ danger')$
      (b.2) $processa('May\ the\ force\ be\ with\ you')$
      (b.3) $processa('Voce\ falou\ em\ pipoca?')$
      (b.4) $processa('Pipoca\ quente\ na\ manteiga')$

3. (2 pontos) Considere o código seguinte:

```
# Comentário omitido
# dict -> dict
def analisa(D):
    R = {}
    for x in D:

        v = D[x]
        if v in R:
            list.append(R[v], x)
        else:
            R[v] = [x]
    return R
```

```
# Teste da função analisa
def teste():
   P = {}
   M = P # M e P referem-se
         # ao mesmo dicionário
   P['Bart'] = 'Homer'
   M['Bart'] = 'Marge'
   P['Stewie'] = 'Peter'
   M['Chris'] = 'Lois'
   M['Lisa'] = 'Lois'
   # A
   P['Lisa'] = 'Homer'
   P['Maggie'] = 'Homer'
   M['Meg'] = 'Lois'
   M['Lisa'] = 'Marge'
   Q = analisa(P)
   # B
   # o restante desta função
   # foi omitido
```

Durante a execução da função teste(), qual é o valor de:

(a) P['Bart'] ao executar os comandos até a linha contendo # A.

(b) P['Lisa'] ao executar os comandos até a linha contendo # A.

(c) Q['Homer'] ao executar os comandos até a linha contendo # B.

(d) Q['Lois'] ao executar os comandos até a linha contendo # B.

(e) Q['Maggie'] ao executar os comandos até a linha contendo # B.

4. (4 pontos) Crie um programa completo (criando função *main*) que gerencie distâncias entre cidades. O programa deve (nesta ordem):

   - pedir ao usuário o número total de cidades, e em seguida pedir o nome de cada cidade;
   - inicializar uma matriz M $n \times n$ preenchida com zeros (crie uma função para isto), onde n é o número total de cidades;
   - pedir ao usuário a distância entre cada par de cidades. Por exemplo, se as cidades informadas foram Maceió, Aracajú e Salvador, o programa deve pedir ao usuário as distâncias entre: Maceió e Aracajú, Maceió e Salvador, Aracajú e Salvador. Estas distâncias devem ser armazenadas na matriz M, de maneira simétrica, ou seja $M[i][j] = M[j][i]$;
   - pedir ao usuário os nomes de duas cidades e imprimir na tela a distância entre elas, (crie uma função para calcular esta distância), ou uma mensagem de erro caso alguma das cidades informadas não esteja na lista de cidades;

   A interação com o usuário deve ser feita apenas na função *main*, isto é, não utilize *input* ou *print* em outras funções. Dica: use a função $list.index(L, x)$ para encontrar o índice do elemento x na lista L.

BOA SORTE !