MAC0122 Princípios de Desenvolvimento de Algoritmos

Bacharelado em Estatística, Matemática e Matemática Aplicada

Terceira Prova – 24 de novembro de 2016

Nome: _____

Assinatura: _____

Nº USP: _____ Prof: _____

Instruções:

1. Não destaque as folhas deste caderno. A prova pode ser feita a lápis.
2. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno está completo.
3. As questões podem ser resolvidas em qualquer página. Ao escrever uma solução (ou parte dela) em página diferente do enunciado, escreva QUESTÃO X em letras ENORMES junto da solução.
4. As soluções devem ser em Python. Você pode usar apenas recursos de Python vistos em aula. Você pode definir funções auxiliares e usá-las à vontade. Cuidado com a legibilidade e, principalmente, com a TABULAÇÃO.
5. As soluções não precisam verificar consistência de dados.
6. Não é permitido o uso de folhas avulsas para rascunho, a consulta a livros, apontamentos, colegas ou equipamentos eletrônicos. Desligue o seu celular e qualquer equipamento que possa perturbar o andamento da prova.

DURAÇÃO PROVA: 2 horas

Questão	Valor	Nota
1	3,0	
2	3,5	
3	3,5	
Total	10,0	

QUESTÃO 1 (vale 3,0 pontos)

Escreva em Python uma função com o seguinte protótipo:

```
def calcule( posfixa ):
  ''' (list) -> float

  Recebe um lista `posfixa` com strings representando uma expressão
  numérica em notação posfixa e retorna o valor da expressão.

  Pré-condição: a função supõe que a expressão está correta.
  '''
```

Cada string na lista `posfixa` pode ser um `'+'`, `'-'`, `'*'`, `'/'` ou um número, como por exemplo `'21'` e `'3.14'`.

Exemplos de execuções da função `calcule()` no Python Shell:

```
>>>
>>> calcule(['1.1'])
1.1
>>> calcule(['1.1', '1.6', '-'])
-0.5
>>> calcule(['1', '2', '3', '-', '+'])
0.0
>>> calcule(['21', '6', '+', '345', '7', '*', '-'])
-2388.0
>>>
>>> calcule(['11', '33', '+', '21', '-'])
23.0
>>>
```

QUESTÃO 2 (vale 3,5 pontos)

Vimos em aula como visualizar o conjunto `Julia(c)` formado pelos pontos de um retângulo do plano (complexo) que não divergem após um certo número de iterações para uma constante complexa `c`.

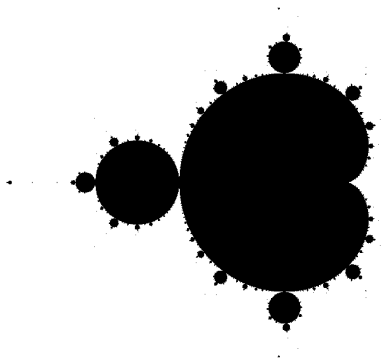
Nessa questão você deve escrever uma função `Mandelbrot()` que cria e retorna a imagem do conjunto de Mandelbrot em um dado retângulo do plano. Cada pixel dessa imagem tem a cor `DENTRO` ou `FORA`. O pixel associado a um ponto complexo `c` tem a cor `FORA` caso a seguinte sequência de $z_0, z_1, \ldots, z_{N\_MAX}$ divirja em até `N_MAX` iterações.

$$z_0 = 0 + 0i$$
$$z_{k+1} = z_k^2 + c, \quad \text{para } k = 1, 2, \ldots, N\_MAX$$

Consideramos que a sequência diverge se em até `N_MAX` iterações o módulo de z_k se torna maior que um dado valor `DELTA`, ou seja, $|z_k| >$ `DELTA`.

O pixel associado a um ponto complexo `c` tem a cor `DENTRO` caso a sequência não divirja.

A figura abaixo ilustra a imagem de resolução `nlin=640` linhas e `ncol=480` colunas, no retângulo definido pelo ponto inferior esquerdo `(hmin, vmin)=(-2.0, -2.0i)` e pelo ponto superior direito `(hmax, vmax)=(2.0, 2.0i)`.



Escreva a função `Mandelbrot()` como descrita abaixo.

```
import numpy as np
## Constantes que você DEVE usar na sua solução
DENTRO = 0    # cor de um ponto que não diverge
FORA   = 255  # cor de um ponto que diverge
N_MAX  = 255  # máximo número de iterações
DELTA  = 10   # limiar

def Mandelbrot(hmin, hmax, vmin, vmax, nlin, ncol):
    '''(float, float, float, float, int, int) -> int array
    Recebe um retângulo complexo de canto inferior esquerdo
    (hmin, vmin) e de canto superior direito (hmax, vmax).
    A função deve retornar um array de inteiros (imagem) de tamanho
    (nlin, ncol), para onde o retângulo deve ser mapeado.
    Os pixels recebem o valor (cor) DENTRO ou FORA de acordo com o enunciado.'''
```

QUESTÃO 3 (vale 3,5 pontos)

Nessa questão, você deve escrever um programa (função `main()`) que use obrigatoriamente objetos das classes `Turtleship` e `Vetor` **sem** implementá-las. A especificações dessas classes estão a seguir.

```
class Vetor:
    def __init__(self, x = 0, y = 0):
        '''(Vetor, int|float, int|float) -> None'''
    def __add__(self, other):
        '''(Vetor, Vetor) -> Vetor'''
    def __sub__(self, other):
        '''(Vetor, Vetor) -> Vetor'''
    def __mul__(self, other):
        '''(Vetor, int|float) -> Vetor
    def __truediv__(self, other):
        '''(Vetor, int|float) -> Vetor
    def distancia(self, other):
        '''(Vetor, Vetor) -> float'''

class Turtleship: ## trata-se de uma simplificação da classe usada no EP8
    def __init__(self, nome, p, v):
        '''(Turtleship, str, Vetor, float) -> None
        '''
        self.nome = nome    # nome da Turtleship
        self.pos  = p       # posição da nave
        self.vel  = v       # velocidade de cruzeiro da nave

    def viagem(self, nova_posicao):
        '''(Turtleship, vet) -> float
           faz a nave viajar até nova_posicao, atualizando a sua
           posição e retorna o tempo necessário para realizar a viagem '''
```

Além das classes `Turtleship` e `Vetor`, você deve utilizar as seguintes constantes:

```
TPAR   =  10 # tempo em minuto
VONI   =  50 # km/min
VTAX   = 100 # km/min
```

Seja `n` um número inteiro positivo e considere uma lista com `n` posições de estações espaciais.

Um ônibus espacial sai da estação de índice zero (primeira estação) e percorre as estações em sequência, até a última estação. O ônibus permanece `TPAR` minutos em cada estação, inclusive a primeira. Portanto ele sai da primeira estação no instante `TPAR`.

Nessa questão você deve escrever um programa que ajude o motorista de um taxi espacial a levar um passageiro a uma estação para pegar o ônibus espacial. O destino do passageiro é a última estação.

O seu programa deve ler:

- um inteiro positivo `n`;
- as `n` posições `(x, y)` das estações espaciais;
- a posição `(x_tax, y_tax)` onde o taxi pega o passageiro;
- o instante `t_tax` do início da corrida do taxi.

O seu programa deve imprimir uma mensagem ao motorista do taxi informando o menor índice de uma estação para onde ele deve conduzir o passageiro para pegar o ônibus.

Cada posição é representada por um objeto da classe `Vetor`. O taxi e o ônibus espaciais são representados por objetos da classe `Turtleship`. O nome do taxi é `'Taxi'` e sua velocidade é `VTAX`. O nome do ônibus é `'Onibus'` e sua velocidade é `VONI`.

Exemplos de execução.

```
Número de estações: 3
x da estação 0: 100
y da estação 0: 0
x da estação 1: 300
y da estação 1: 0
x da estação 2: 700
y da estação 2: 0
x do passageiro: 200
y do passageiro: 0
Início da corrid: 8
Siga para:  0
```

```
Número de estações: 3
x da estação 0: 100
y da estação 0: 0
x da estação 1: 300
y da estação 1: 0
x da estação 2: 700
y da estação 2: 0
x do passageiro: 200
y do passageiro: 0
Início da corrida: 12
Siga para:  1
```

```
Número de estações: 3
x da estação 0: 100
y da estação 0: 0
x da estação 1: 300
y da estação 1: 0
x da estação 2: 700
y da estação 2: 0
x do passageiro: 200
y do passageiro: 0
Início da corrida: 30
Siga para:  2
```