MAC2166 – Introdução à Computação para Engenharia

Escola Politécnica

Segunda Prova – 13 de maio de 2013

Nome: _____

Assinatura: _____

Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade e, principalmente, com a TABULAÇÃO.
4. Qualquer questão pode ser resolvida em qualquer página. Se a questão não está na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO X em letras ENORMES antes da solução.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de equipamentos eletrônicos.
8. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1	3,0	
2	3,0	
3	4,0	
Total	10,0	

Questão 1 (vale 3 pontos)

Esta questão é composta por 3 itens. Assuma que as funções e programa a serem escritos estão todos em um mesmo arquivo, sem a necessidade de importação de módulos.

item (a) (vale 1 ponto) Para um inteiro $r \geq 0$, denote por K_r o número

$$1 - \frac{1}{3} + \frac{1}{5} - \cdots + (-1)^r \frac{1}{2r+1}.$$

Assim, por exemplo, $K_0 = 1$, $K_1 = \frac{2}{3} = 0{,}6666\ldots$, $K_2 = \frac{13}{15} = 0{,}8666\ldots$ e $K_3 = \frac{76}{105} = 0{,}7238\ldots$.

Escreva uma função de cabeçalho

```
def K(r):
```

que recebe como parâmetro um inteiro $r \geq 0$ e retorna o valor de K_r.

item (b) (vale 1,5 pontos) Considere a função, definida para todo x com $|x| < 1$, dada pela série

$$f(x) \;=\; \frac{x^2}{2}K_0 + \frac{x^6}{6}K_1 + \frac{x^{10}}{10}K_2 + \ldots + \frac{x^{4r+2}}{4r+2}K_r + \ldots$$

Escreva uma função de cabeçalho

```
def f(x,eps):
```

que recebe como parâmetro um real `x` tal que $|$`x`$| < 1$ e calcula uma aproximação de $f(x)$ dada pela série acima, incluindo na soma todos os termos até o primeiro de valor menor que `eps`. O primeiro termo de valor menor que `eps` **DEVE** ser incluído na soma. A função deve retornar o valor da aproximação e o valor do último termo incluído na aproximação, nesta ordem. Utilize **obrigatoriamente** a função do item anterior, mesmo que você não a tenha feito.

item (c) (vale 0,5 ponto) Escreva um programa que lê um número real `x`, com $|\texttt{x}| < 1$, um real `eps` > 0, e imprime o valor da aproximação de $f(x)$ e o último termo incluído na aproximação. Utilize **obrigatoriamente** a função do item anterior, mesmo que você não a tenha feito. O valor de `eps` deve ser usado na chamada dessa função.

Questão 2 (vale 3 pontos)

Escreva um programa que lê

- um inteiro positivo `m`,
- uma sequência de `m` números reais,
- um inteiro positivo `n` e
- uma sequência de `n` números reais.

O programa deve imprimir os números que aparecem nas duas sequências.

Por exemplo, para entrada

```
6
4.1  -7.2  5.0  24.2  5.0  8
4
5.0   6.3  5.0  -7.2
```

o programa deve imprimir

```
O número 5.0 aparece nas duas sequências.
O número -7.2 aparece nas duas sequências.
```

Note que o número 5, que ocorre várias vezes nas duas sequências, deve ser impresso apenas uma vez.

Não se preocupe com o formato em que os números serão impressos.

Caso ache conveniente, escreva (e use) funções adicionais.

Questão 3 (vale 4 pontos)

Esta questão é baseada no EP3. Suponha que todas as funções dadas e as que você precisa escrever estão em um mesmo arquivo — para usar essas funções não há necessidade de importação de módulos.

item (a) (vale 1 ponto) Escreva uma função com o cabeçalho

```
def atualize_turtleship(turtleship, lista_astros, delta_t):
```

que recebe:

- uma referência a uma turtleship através de ´`turtleship`´,
- uma referência a uma lista de astros através de ´`lista_astros`´
- um intervalo de tempo ´`delta_t`´.

Uma turtleship é representada através de uma lista que tem a forma `[[x_t,y_t], [vx_t,vy_t], cor_t, nome_t, ativa_t]`, onde

- `[x_t,y_t]` é a posição da turtleship em um instante `t`,
- `[vx_t,vy_t]` é a velocidade da turtleship em um instante `t`,
- `cor_t` é a cor da turtleship,
- `nome_t` é o nome da turtleship,
- `ativa_t` indica se a turtleship está operacional e inicialmente é `True`.

Esta função deve calcular a posição e velocidade da turtleship no instante `t+delta_t` e atualizá-las.

Dados a posição x, a velocidade v e a aceleração a num instante t, podemos calcular seus valores x', v' no instante seguinte $t + \Delta t$ pelas fórmulas:

$$\begin{aligned} x' &= x + v\Delta t + a\Delta t^2/2, \\ v' &= v + a\Delta t. \end{aligned}$$

Na sua função, você **deve** usar a função a seguir **sem escrevê-la** (suponha que ela é dada):

```
def aceleração_resultante(lista_astros, ponto):
```

que recebe em `lista_astros` uma lista de astros celestes e um `ponto=[x,y]` e retorna o vetor aceleração `[a_x,a_y]` da força gravitacional exercida pelos astros sobre um objeto no ponto.

item (b) (vale 3 pontos) Escreva uma função com o cabeçalho

```
def simule(lista_astros, lista_turtleships, t_max, delta_t, d_colisão):
```

que recebe:

- uma lista de astros `lista_astros`,
- uma lista com 2 turtleships `lista_turtleships`,
- o tempo máximo `t_max` de simulação,
- intervalo de tempo `delta_t`,
- uma distância de colisão `d_colisão`,

e simula as trajetórias das 2 turtleships em `lista_turtleships` sob o efeito da força gravitacional exercida pelos astros em `lista_astros`.

O instante inicial da simulação é 0 e a cada passo da simulação o tempo de simulação é acrescido de `delta_t`. Uma turtleship colide com a outra se a distância entre elas for menor do que `d_colisão`. Caso haja colisão, imprima a seguinte mensagem: "Colisão entre as tartarugas!"

Uma turtleship será desativada se ela colidir com algum astro ou se ela **colidir com a outra turtleship**. No caso de colisão com um astro, o tempo da simulação deverá ser impresso. Havendo colisão simultânea, entre as turtleships e entre uma turtleship e um astro, somente a colisão entre as turtleships deve ser informada.

As posições das turtleships ativas deverão ser atualizadas e impressas em cada passo da simulação.

A simulação termina quando o tempo de simulação ultrapassar `t_max` ou quando não houver nenhuma turtleship ativa (todas turtleships colidiram). Imprima uma mensagem indicando o motivo do término da simulação.

Na sua função, você **deve** usar as funções a seguir **sem escrevê-las** (suponha que elas são dadas) :

```
def distância(ponto_1, ponto_2):
```

que recebe dois pontos `ponto_1=[x1,y1]` e `ponto_2=[x2,y2]` e retorna a distância entre `ponto_1` e `ponto_2`.

```
def houve_colisão(turtleship, lista_astros):
```

que recebe uma `turtleship` e uma `lista_astros` e devolve `True` se a turtleship colidiu com algum astro e, em caso contrário, retorna `False`.

O uso das funções da biblioteca matemática é livre. Aqui vai uma pequena lista delas: `cos(x)`, `sin(x)`, `hypot(x,y)`, `pow(x,y)`, `sqrt(x)`, `tan(x)`, `acos(x)`, `asin(x)` e `atan(x)`. Caso use alguma dessas funções, não esqueça de importar o módulo matemático.