LISTA DE EXERCÍCIOS 3 - PROGRAMAÇÃO 1

PROF. FERNANDO NETO

SISTEMAS DE INFORMAÇÃO – CIN – UFPE

2017.2

Observação inicial: Para ambas as questões desta lista, você deve documentar seu arquivo com um cabeçalho descritivo usando docstring, como nesse exemplo. Isso será avaliado.

1) TRIVIA

Faça um programa que execute um jogo de perguntas e respostas utilizando duas listas, uma com as perguntas e outra com as respostas, cada pergunta deve ter sua resposta e ambas precisam estar ordenadas, ou seja, cada elemento da lista de perguntas deve ter um elemento correspondente na lista de respostas(ambos no mesmo índice de cada lista).

Exemplo: listaPerguntas = [pergunta1, pergunta2, pergunta3]

listaRespostas = [resposta1, resposta2, resposta3]

Formato de entrada:

A entrada será uma pergunta seguida de uma resposta, o usuário pode então inserir quantas questões forem necessárias, caso ele não coloque nada no espaço da próxima pergunta, o programa deve parar e iniciar o jogo, caso ele não coloque nada no espaço da resposta, o jogo se inicia mas sem a última pergunta inserida. **Você precisa se certificar que todas as perguntas têm respostas correspondentes.**

Formato de saída:

Após inserir todas as questões e iniciar o programa, ele deve começar o jogo. O programa fará a primeira pergunta e pedir um input do usuário, que deverá inserir a resposta. Depois disso ele deve retornar a pontuação (onde cada resposta correta adiciona um ponto ao total) e as perguntas que ele acertou ou não (para isso você pode criar uma terceira lista com o mesmo tamanho da lista de perguntas/respostas com valores booleanos).

Exemplos:

Obs. Trechos em vermelho significam input do usuário.

Exemplo 1:

Parte 1: Inserir perguntas e suas respectivas respostas (aqui o usuário coloca quantas questões ele desejar, caso não coloque nada, o jogo se inicia.)

```
>>>Em Harry Potter, quantos pontos valem a captura do pomo de ouro?
>>>150
>>>Em o Senhor dos Anéis, o que é "Leve como uma pena, e dura como escamas de
```

```
dragão”?
>>>mithril
>>>
```

Parte 2: Início do jogo (aqui o programa retorna as perguntas e o usuário tenta responder as questões).

```
Em Harry Potter, quantos pontos valem a captura do pomo de ouro? 120
Em o Senhor dos Anéis, o que é "Leve como uma pena, e dura como escamas de
dragão”? mithril
```

Parte 3: Resultado (assim que as perguntas se acabarem, o jogo mostra o resultado)

```
Resultado:
Pergunta 1: Resposta Errada 120 (150)
Pergunta 2:Resposta Certa (mithril)
Pontuação: 1/2
```

Exemplo 2:

Parte 1: Inserir perguntas e suas respectivas respostas (caso o usuário inicie o programa mas logo em seguida não adicione nada, o jogo não começa)

```
>>“”
Não há perguntas a serem feitas.
```

Exemplo 3:

Parte 1: Inserir perguntas perguntas e suas respectivas respostas

```
>>Quantos socos sérios são necessários para Saitama derrotar seu inimigo?
>>1
>>Qual seria o output da seguinte operação em python: "a" + "bc"(em caso de erro
responda "erro")?
>>abc
>>Quando será lançado Half Life 3?
>> “”
```

Parte 2:Início do Jogo

“Quantos socos sérios são necessários para Saitama derrotar seu inimigo?”1 “Qual seria o output da seguinte operação em python: "a" + "bc"(em caso de erro responda "erro").”abc

Parte 3: Resultado

Resultado: Pergunta 1: Resposta Certa (1) Pergunta 2: Resposta Certa (abc) pontuação:2/2

2) REVERTENDO A ENTROPIA

Motivação:

Ordenar uma lista de coisas é algo que fazemos desde de criança e é parte fundamental da nossa forma de viver, pensar e se relacionar com o mundo. **Ordenamos nossos nomes na sala, ordenamos livros nas bibliotecas, ordenamos músicas no spotify.** Na cadeira de algoritmos, do segundo período, vocês verão vários dos chamados ***algoritmos de ordenação*** e como eles podem ser mais ou menos eficientes uns que os outros, dependendo da situação. Porém, como não estamos preocupados com isso aqui em P1, construiremos um deles de forma mais lúdica possível.

Objetivo:

Nosso objetivo geral nessa questão será receber uma lista de números desordenados do usuário, e imprimi-los em ordem crescente.

Diferente de quaisquer outras questões que você já tenha experienciado nas listas anteriores, essa terá várias etapas que deverão ser seguidas, e cada uma terá sua própria saída na tela. Portanto organize seu código para não se perder.

Formato de entrada:

A entrada será dada na **primeira etapa**, e serão apenas **números inteiros** (suponha que o usuário não tentará “quebrar” seu programa”)**.** O programa deve receber números inteiros até o usuário digitar qualquer coisa que não seja um número inteiro.

Etapas:

A. Receba do usuário quantos números inteiros ele digitar, até que ele digite qualquer coisa que não seja um número inteiro. **Guarde todos os números em uma lista chamada "original".** Não se esqueça de fazer as verificações e conversões necessárias.
   a. Para essa questão, imprima na tela **"a) " + original.**
   b. Para o caso especial onde a lista é vazia pois a primeira entrada do usuário já não foi um número inteiro, imprima "Sua lista de números está vazia.", ***e encerre o programa.***

B. Agora, verifique se essa lista já está **desordenada**. Para fazer isso você deve procurar por **um caso de desordenação**, que seria por exemplo: [1,2,3,**5,4**,6,**9,8**].

   Se em qualquer momento da lista, existe um (índice) e um (índice + 1) tal que o valor do (índice) é maior que o valor do (índice+1), ***é porque a lista está desordenada****.*

   Lembre-se isso é uma etapa de verificação, você **não deve** alterar a lista.
   a. Para essa etapa, imprima na tela **"b) " + True ou False**, True caso você tenha achado uma desordenação, False caso não.
   b. Lembre-se que uma lista com apenas um elemento está sempre ordenada.

C. Faça uma cópia da sua lista "original" (copie a lista e não seu endereço, clique para mais explicações.). Com **a cópia** em mãos você fará novamente **uma única varredura** por sua lista procurando casos de desordenação (como descrito na questão anterior). Porém, dessa vez você alterará a cópia da lista: Para cada caso que você achar, você trocará de posição o que há no índice atual, com o próximo. E seguirá assim até o final da lista. Segue as etapas para exemplo na questão anterior:

[1,2,3,5,4,6,9,8] - **Um > Dois**? Não, testamos a próxima dupla.

[1,2,3,5,4,6,9,8] - **Dois > Três**? Não, testamos a próxima dupla.

[1,2,3,5,4,6,9,8] - **Três > Cinco**? Não, testamos a próxima dupla.

[1,2,3,**5,4**,6,9,8] - **Cinco > Quatro**? Sim!! Vamos trocar esses de posição!

[1,2,3,**4,5**,6,9,8] - Feita a troca, vamos para a próxima dupla.

[1,2,3,4,5,6,9,8] - **Cinco > Seis**? Não, testamos a próxima dupla.

[1,2,3,4,5,6,9,8] - **Seis > Nove**? Não, testamos a próxima dupla.

[1,2,3,4,5,6,**9,8**] - **Nove > Oito**? Sim!! Vamos trocar esses de posição!

[1,2,3,4,5,6,**8,9**] - Feita a troca, iríamos para próxima dupla, mas não há mais duplas, saímos do laço.

a. Para essa etapa, imprima na tela **“c) ” + cópia alterada**.
b. *Lembre-se que uma lista com apenas um elemento está sempre ordenada.*

D. Para finalizar seu algoritmo de ordenação, você deve mesclar o conceito aprendido na etapa B com o da etapa C.

Na etapa B, você aprendeu como verificar se sua lista está ou não ordenada. Você faz isso procurando por um caso de desordenação, afinal se houver um caso de desordenação na sua lista, ela não está ordenada. E obviamente, **se não houver um caso de desordenação na lista, ela está ordenada.**

Na etapa C, você aprendeu a fazer uma micro-ordenação. O que isso quer dizer? Você fez alterações em sua lista que torna ela **mais próxima de uma lista ordenada**, pois você corrigiu as desordenações que você achou. Porém uma única micro-ordenação não garante que a lista termine ordenada para todos os casos.

A idéia para a **etapa D,** portanto, é fazer um código que execute micro-ordenações até o momento que não haja mais nenhum caso de desordenação na lista.

a. Para essa etapa, imprima na tela **“d) ” + lista original ordenada**.
b. *Lembre-se que uma lista com apenas um elemento está sempre ordenada.*

Formato de saída:

Ao final, o programa deverá imprimir na tela as quatro as respostas corretas para as quatro letras.

Exemplos:

Entrada	Saída
5	“a) [5,1,2,4,3]” “b) True” “c) [1,2,4,3,5]” “d) [1,2,3,4,5]”
1	
2	
4	
3	
“camelo”	

Entrada	Saída
1	“a) [1,2,3]” “b) False” “c) [1,2,3]” “d) [1,2,3]”
2	
3	
“lhama”	

Entrada	Saída
4.5	“Sua lista de números está vazia.”

Sucesso!