Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Data: 10 de Outubro de 2019
Professora: Elloá B. Guedes
Disciplina: Fundamentos Teóricos da Computação
Monitor: Diego Lucena

Projeto Prático II
Casamento Sensível ao Contexto

O uso de parênteses, colchetes e chaves em expressões aritméticas organiza e facilita o entendimento das mesmas, além de auxiliar na determinação da ordem em que determinadas operações devem ser realizadas. Eliminando-se os números e os operadores, diz-se que uma string contendo uma expressão é considerada casada e correta quando há o casamento do símbolo de abertura com seu símbolo de fechamento correspondente, respeitando o fato de que { e } devem ser mais externos aos [e] e que, por sua vez, devem ser mais externo aos (e). Por exemplo, `{ [ ] [( ) ( )]}` representa uma string casada e correta, `( { [ ] } )` representa uma string casada e incorreta e `( ) ( ( )` representa uma string não casada.

As strings casadas e corretas no tocante ao uso de parênteses, chaves e colchetes compõem uma linguagem sensível ao contexto. No tocante apenas ao casamento, sem considerar a ordem em que os parênteses, chaves e colchetes aparecem, tem-se que esta é uma linguagem livre de contexto. Para esta última, em especial, há um autômato finito não-determinístico com pilha que a reconhece. Assim, usando os conceitos em questão e fazendo uso de uma pilha, você deve determinar quando uma string é ou não casada. Em caso afirmativo, verifique se a string é correta ou não. Quando esta for incorreta, faça a substituição repetitiva de símbolos até que esta se torne correta, mostrando a versão resultante ao final. Neste processo de substituição, você deve aplicar as regras sensíveis ao contexto adequadas como, por exemplo, ([→ [(. No caso das strings não casadas, apenas forneça esta informação como saída.

De maneira sintética, as entradas e saídas do seu problema consistem em:

1. **Entrada**. Várias strings, a serem lidas da entrada padrão, até que uma string vazia seja recebida. Cada string é composta pelos símbolos `[,],(,),{,}` ou espaço em branco.

2. **Saídas**. Para cada linha da entrada, retornar: `casada e correta`, `nao casada` e `casada e incorreta`. Para as strings que incorrem neste último caso, também apresentar ao final a versão casada e correta das mesmas. É importante ressaltar que todos os espaços em branco devem ser ignorados antes da checagem especificada e também nas saídas casadas e corretas após o conserto. Se após a remoção dos espaços restar apenas uma string vazia, tem-se que esta é casada e correta por vacuidade.

Para fins de simplificação, sempre há uma única versão correta das strings que precisarem de conserto. Entradas como `([][][][])` que podem ensejar saídas como `()[][][][]` ou também `[][][][]()` não serão fornecidas como entrada.

Para resolver o problema em questão, você deve utilizar a linguagem de programação Python 3 e obrigatoriamente fazer uso da estrutura de dados pilha, a qual também pode ser implementada com o auxílio de uma lista, ficando à critério de cada aluno conforme preferir. Os exemplos a seguir auxiliam a ilustrar entradas e saídas para o problema considerado.

1 Exemplos de Entradas e Saídas

Entrada	Saída	Comentário
`[{()}()[]]{}(`	`nao casada`	
`()()()())`	`nao casada`	
	`casada e correta`	String composta apenas de espaços em branco
`[()]{[]}`	`casada e correta`	
`[{}]`	`casada e incorreta`	
`([{}])`	`casada e incorreta`	
`([{()}])`	`casada e incorreta`	
`{[()]([])}`	`casada e incorreta`	
	`{[]}`	Entrada vazia
	`{[()]}`	
	`{[(())]}`	
	`{[()][()]}`	

2 Observações Importantes

- Lembre-se, a entrada de dados é feita via `input` e a saída via `print`;
- Atenha-se exatamente ao padrão de entrada e saída fornecidos nos exemplos. Qualquer mensagem adicional na entrada ou na saída de dados pode culminar em incorretude;
- Cuidado ao copiar caracteres do PDF! Eles podem estar com codificação incorreta. Atente-se ao enunciado;
- Na construção do seu programa você deve usar apenas os conceitos aprendidos em sala de aula. Respostas que utilizem bibliotecas prontas não serão consideradas;
- Em caso de plágio, todos os envolvidos receberão nota zero!
- Na execução do seu programa no *run.codes*, existem casos de testes que vão além dos exemplos mostrados a seguir. Esses casos de teste não serão revelados. Pense em exemplos de entradas e saídas que podem acontecer e melhore o seu código para capturá-las.

3 Prazos Importantes

- **Início**. 10/10/2019 às 16h20min (horário do servidor)
- **Encerramento**. 23/08/2019 às 23h55min (horário do servidor)

4 Links Úteis

- https://docs.python.org/3.1/tutorial/datastructures.html
- http://openbookproject.net/thinkcs/python/english3e/stacks.html