Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Data: 23 de Abril de 2019
Professora: Elloá B. Guedes
Disciplina: Fundamentos Teóricos da Computação
Monitor: David Yonekura

Projeto Prático II
Casamento de Parênteses

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 bem formada se há o casamento do símbolo de abertura com seu símbolo de fechamento correspondente. Por exemplo, como no caso, "[( )]". Por outro lado, o exemplo '( ]' mostra uma string mal formada.

Na Matemática, em particular, o símbolo '{' deve preceder o símbolo '[' que, por sua vez, precede o símbolo '('. Neste problema, para fins de simplificação, esta precedência não será considerada. Portanto, uma expressão como "[{ }]([ ])" também é considerada bem formada.

As strings bem formadas no tocante ao uso de parênteses, chaves e colchetes compõem uma linguagem que é livre de contexto. Portanto, existe uma gramática livre de contexto que gera esta linguagem e também 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 bem formada. Além disso, suponha ainda que você está analisando dados de um Matemático estudioso, cujas expressões podem exceder uma linha individual. Neste caso, além de checar a formação das linhas individuais, você também deve checar a boa ou má formação da string resultante da concatenação das linhas individuais.

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 `True` caso seja bem formada em relação aos parênteses, chaves ou colchetes e `False`, em caso contrário. É importante ressaltar que todos os espaços em branco devem ser ignorados antes da checagem especificada. Se restar apenas uma string vazia após estas remoções, tem-se que esta é bem formada por vacuidade. Além disso, retornar `True` ou `False` acerca da boa ou má formação da string mais geral.

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
`[{()}()[]]{}(`	`False`	
`()()()())`	`False`	
	`True`	String composta apenas de espaços em branco
`[(){{[]}}]`	`True`	
	`True`	String vazia de entrada
`{[][][][]`	`False`	
`{()()()()(`	`False`	
`{ } { } { } { } { }`	`True`	
	`False`	String vazia de entrada
	`True`	String composta apenas de espaços em branco
	`True`	String composta apenas de espaços em branco
	`True`	String vazia de entrada

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;
- A cada execução do programa será fornecida apenas uma entrada, cujo resultado deve ser exibido ao final do processamento;
- 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**. 23/04/2019 às 13h (horário do servidor)
- **Encerramento**. 30/04/2019 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