Exercício-Programa: Implementação do algoritmo CYK para Gramáticas Livres do Contexto

Objetivo:

Desenvolver um programa para processamento de Gramáticas Livres do Contexto (GLCs). Dada a especificação de uma GLC G na Forma Normal de Chomsky e uma cadeia $w \in \Sigma^*$, seu programa deve determinar se G gera w, e devolver a matriz *Tabela* (vide descrição do algoritmo) e o status (aceita / rejeita). Em outras palavras, o problema é decidir se w pertence a $L(G)$. O programa deverá ser desenvolvido em C, Python ou Java, e deverá ser executável via linha de comando do terminal.

Entrada:

A chamada do programa será feita através de linha de comando, pelo nome:

Implementação em C:

```
glc.exe
```

Implementação em Java:

```
java glc
```

Implementação em Python:

```
py glc.py
```

O programa terá os seguintes arquivos de entrada:

- inp-glc.txt:
  Conterá a especificação da GLC (na Forma Normal de Chomsky).

  A primeira linha conterá o campo:
  n
  no qual *n* é o número de gramáticas a serem testadas

A segunda linha conterá os campos:

$q_1\ t_1\ s_1$

onde q_1 é o número de variáveis, t_1 é o número de símbolos terminais e s_1 é o número de regras de substituição referentes à gramática G_1. Note que todos esses parâmetros devem ser estritamente positivos.

A terceira linha conterá a lista de variáveis da gramática G_1, separadas por espaços. A variável inicial deve aparecer em primeiro lugar na lista.

A quarta linha conterá a lista de terminais da gramática G_1, separados por espaços.

As linhas 5 a 5 + s_1 - 1 conterão as regras de substituição, em uma das formas abaixo:
Variável => &
Variável => terminal
Variável => Variável Variável

As linhas 5 + s_1 adiante conterão as especificações da próxima gramática, e assim por diante até o fim da especificação da gramática G*n*.

Observações:

- Nas regras de substituição, as variáveis e símbolos serão separados por espaços;

- O símbolo reservado & representará cadeia vazia.

O exemplo a seguir ilustra a especificação para duas GLCs.

Primeira GLC:

$S_0 \rightarrow \& \mid A\ T \mid B\ U \mid S\ S \mid A\ B \mid B\ A$
$S \rightarrow A\ T \mid B\ U \mid S\ S \mid A\ B \mid B\ A$
$T \rightarrow S\ B$
$U \rightarrow S\ A$
$A \rightarrow a$
$B \rightarrow b$

Segunda GLC:

$S_1 \rightarrow AB$

$A \rightarrow 0$

$B \rightarrow 1$

Arquivo inp-glc.txt correspondente a essas duas gramáticas:

```
2
6 2 15
S0 S T U A B
a b
S0 => &
S0 => A T
S0 => B U
S0 => S S
S0 => A B
S0 => B A
S => A T
S => B U
S => S S
S => A B
S => B A
T => S B
U => S A
A => a
B => b
3 2 3
S1 A B
0 1
S1 => AB
A => 0
B => 1
```

- inp-cadeias.txt:

Conterá as cadeias a serem avaliadas.

A primeira linha conterá um inteiro n_1 representando o número de cadeias a serem

processadas pela gramática G_1.

As $n_1$linhas conterão uma cadeia, onde os símbolos serão separados por espaços.

A linha n_1+1conterá o inteiro n_2, indicando o número de cadeias a serem processadas pela gramática G_2. E assim por diante até o inteiro n_k, onde k é o número de gramáticas de teste.

Ex:

3
&
a
a b a a b b
3
&
0 1
0 0 1

Os arquivos de saída serão:

- out-status.txt:
  Conterá *n* linhas, sendo a linha *i* referente aos status (0 = rejeita, 1 = aceita) das cadeias de entrada segundo a gramática G_i. Esses dígitos serão separados por espaços.

  Para o exemplo anterior da primeira GLC, o arquivo conteria as linhas:

  1 0 1
  0 1 0

Entrega do trabalho:

Condições da entrega:

- O trabalho poderá ser feito em grupos de ATÉ dois alunos, devidamente identificados na primeira linha do código-fonte.
- **O prazo para entrega é 16/11/2020.**

- Deverá ser entregue um diretório compactado (formato .zip) contendo o arquivo fonte e o executável (ou classe Java compilada). O diretório deve ser nomeado na forma *d<numerousp1> <numerousp2>.zip*. O módulo principal deve ter o nome glc.c, glc.java ou glc.py.

- O código-fonte deverá ser compilável via comando gcc, javac ou py. Se desenvolver seu programa em IDEs como Eclipse ou Netbeans, certifique-se de que seu programa seja compilável sob as condições aqui expostas.

- Inclua também, no diretório de seu ep, um arquivo chamado LEIAME.TXT contendo o comando de chamada do compilador (gcc, javac ou py) e os argumentos necessários para compilação. Em outras palavras, o arquivo LEIAME.TXT deverá conter a linha de comando exata a ser digitada no prompt do sistema operacional para compilar seu programa.

- O trabalho deverá ser submetido via E-DISCIPLINAS na atividade EP2. Na submissão No corpo da mensagem e nas 1as linhas do código fonte deverá constar os nomes e números USP dos membros do grupo.

  Não é necessário que os dois alunos do grupo enviem o código-fonte, basta um e-mail por grupo.

- Dúvidas a respeito das especificações ou a respeito da implementação do trabalho serão sanadas até o dia 13/11. Dúvidas encaminhadas após este prazo serão ignoradas.

- Além da correção do programa, será considerada a qualidade da documentação do código fonte.

- Se houver evidência de plágio entre trabalhos de grupos distintos, os mesmos serão desconsiderados.