Teoria da Computação

2022/2 - **Trabalho da disciplina**

Enunciado

Implemente em C/C++ um programa que lê do usuário uma Gramática Livre de Contexto na Forma Normal de Chomsky e uma sequência de palavras e, para cada palavra, determina se ela pertence ou não à linguagem gerada pela gramática.

Entrada

A primeira linha da entrada conterá um inteiro P, o número de produções da gramática. Cada uma das próximas P linhas conterá uma produção, na forma A `->` BC ou A `->` a, onde A, B e C são símbolos não terminais, e a é símbolo terminal.

Símbolos não terminais serão sempre dados por letras maiúsculas do alfabeto (`A`, `B`, ..., `Z`), e símbolos terminais serão dados por outros caracteres. O símbolo inicial será sempre dado pela letra `S`. A gramática dada sempre estará na Forma Normal de Chomsky, e ela nunca gera a palavra vazia ε.

Em seguida, a entrada conterá uma sequência de palavras, uma por linha. Para cada palavra, seu programa deverá determinar se ela pertence ou não à linguagem gerada pela gramática dada. As palavras poderão conter quaisquer caracteres, exceto espaços em branco.

A entrada termina com uma linha contendo apenas `*`.

Saida

Para cada palavra p, imprima uma linha contendo p`: SIM` se p pertence à linguagem, ou p`: NAO` caso contrário.

Exemplo

A gramática dada no exemplo de entrada e saida abaixo gera a linguagem $\{a^n b^n \mid n \geq 1\}$:

Exemplo de entrada	**Exemplo de saida**

`7`
`S -> AX`
`S -> AB`
`R -> AX`
`X -> RB`
`R -> AB`
`A -> a`
`B -> b`
`aaabbb`
`aababb`
`aabb`
`aaabb`
`aabba`
`abb`
`ab`
`*`
	`aaabbb: SIM`
	`aababb: NAO`
	`aabb: SIM`
	`aaabb: NAO`
	`aabba: NAO`
	`abb: NAO`
	`ab: SIM`

Outros exemplos são dados ao final deste documento.

Implementação

- O trabalho deve ser feito em C ou em C++;
- Você pode utilizar o algoritmo que preferir para a implementação do trabalho. Suas opções incluem, mas não são limitadas a:
  - implementar um autômato à pilha (PDA), converter produções da gramática para transições do PDA, e simulá-lo em seguida;
  - buscar uma derivação válida através de função(es) recursiva(s) do tipo `bool produz(S, i, j)`, que retorna "verdadeiro" se é possível produzir a (sub)string `palavra[i..j]` a partir do símbolo não terminal `S`, ou "falso" caso contrário;
  - implementar o algoritmo CYK[1];
  - outras idéias.
- O tempo de execução da sua solução não será levado em consideração na correção do trabalho. Entretanto, seu programa deve terminar em tempo "razoável" (< 1 min) para os exemplos dados neste documento;
- O trabalho poderá ganhar até 20 *pontos extras* (e valer ao todo 120 pontos) com as seguintes funcionalidades extras:
  - (10 pontos extras) para cada palavra dada que pertence à linguagem gerada pela gramática, imprimir sua derivação (como sequência ou como árvore). Veja exemplos no final deste documento;
  - (10 pontos extras, **desafio**) funcionar para gramáticas no geral (que não estão na Forma Normal de Chomsky). Novamente, você pode usar o algoritmo que preferir, incluindo, mas não limitado a, transformar a gramática na Forma Normal de Chomsky e então aplicar a solução desenvolvida anteriormente. Veja exemplos no final deste documento.

Orientações

- O trabalho pode ser feito por equipes de *até* 2 (dois) estudantes;
- Submeta, via *Moodle*, um pacote (zip ou tar.gz) contendo todo o código-fonte do trabalho, além de um arquivo de texto (txt) onde conste:
  - O nome de todos os integrantes da equipe;
  - Toda informação que a equipe julgar relevante para a correção (como *bugs* conhecidos, detalhes de implementação, escolhas de projeto, etc.)
- Comente adequadamente seus códigos para facilitar a correção.
- Atenção: a correção será parcialmente automatizada, e a saída do programa será testada com outras entradas além das fornecidas como exemplo. *Siga **fielmente** o formato de saída dado nos exemplos*, sob pena de grande redução da nota;
- Certifique-se que seu programa compila e funciona antes de submetê-lo;
- O trabalho deve ser entregue até **6 de Novembro de 2022, 23:59**, apenas via *Moodle*. Trabalhos entregues por outros meios ou fora do prazo não serão aceitos. É suficiente que o trabalho seja submetido por apenas um estudante da equipe;
- Trabalhos detectados como cópia, plágio ou comprados receberão **todos** a nota 0 (**ZERO**) e estarão sujeitos a abertura de Processo Administrativo Disciplinar Discente.

[1] `https://en.wikipedia.org/wiki/CYK_algorithm`

A gramática abaixo gera expressões aritméticas cujos operandos são números de apenas um dígito:

Exemplo de entrada	**Exemplo de saida**
41 S -> AX S -> RY S -> RZ S -> RW S -> RP S -> 0 S -> 1 S -> 2 S -> 3 S -> 4 S -> 5 S -> 6 S -> 7 S -> 8 S -> 9 R -> AX R -> RY R -> RZ R -> RW R -> RP R -> 0 R -> 1 R -> 2 R -> 3 R -> 4 R -> 5 R -> 6 R -> 7 R -> 8 R -> 9 X -> RF Y -> MR Z -> BR W -> VR P -> DR A -> (F ->) M -> + B -> - V -> * D -> / 2*(3+5) 2*)3+5(2*(3+x) (5/0)-(4*9) 2*(0/(4+5)) 6*(3/(7-6) 8*2/(8*1) 9*1/2-3) putz *	2*(3+5): SIM 2*)3+5(: NAO 2*(3+x): NAO (5/0)-(4*9): SIM 2*(0/(4+5)): SIM 6*(3/(7-6): NAO 8*2/(8*1): SIM 9*1/2-3): NAO putz: NAO

A gramática abaixo gera expressões regulares sobre $\Sigma = \{a, b, c\}$:

Exemplo de entrada	**Exemplo de saida**
20 S -> a S -> b S -> c S -> RX S -> RA S -> RR S -> OY R -> a R -> b R -> c R -> RX X -> MR R -> RA R -> RR R -> OY Y -> RC O -> (C ->) A -> * M -> + abba (a+b)* (a+)* (a+b)*cac(a+b)* a*b*a *a a* abba+* abca+b* a(b+c) *	abba: SIM (a+b)*: SIM (a+)*: NAO (a+b)*cac(a+b)*: SIM a*b*a: SIM *a: NAO a*: SIM abba+*: NAO abca+b*: SIM a(b+c): SIM

No exemplo abaixo, a saida apresenta também a derivação das palavras que pertencem à linguagem (o que vale até 10 pontos extras):

Exemplo de entrada

```
7
S -> AX
S -> AB
R -> AX
X -> RB
R -> AB
A -> a
B -> b
aaabbb
aababb
aabb
aaabb
aabba
abb
ab
*
```

Exemplo de saida

```
aaabbb: SIM
S
AX
aRB
aAXb
aaRBb
aaABbb
aaabbb

aababb: NAO
aabb: SIM
S
AX
aRB
aABb
aabb

aaabb: NAO
aabba: NAO
abb: NAO
ab: SIM
S
AB
ab
```

No exemplo abaixo, o programa reconhece gramáticas fora da Forma Normal de Chomsky (o que vale até 10 pontos extras):

Exemplo de entrada

```
26
S -> S+S
S -> S-S
S -> S*S
S -> S/S
S -> (S)
S -> N
N -> 0
N -> 1
N -> 2
N -> 3
N -> 4
N -> 5
N -> 6
N -> 7
N -> 8
N -> 9
N -> N0
N -> N1
N -> N2
N -> N3
N -> N4
N -> N5
N -> N6
N -> N7
N -> N8
N -> N9
42
1024
(85+32)*(5/4)-3
(20*(3+5)
(-3+4)/2
(0-3+4)/2
*
```

Exemplo de saida

```
42: SIM
1024: SIM
(85+32)*(5/4)-3: SIM
(20*(3+5): NAO
(-3+4)/2: NAO
(0-3+4)/2: SIM
```