Problem E

Forma Normal de Chomsky

O objectivo deste problema é a implementação dum algoritmo de transformação de gramáticas livres de contexto para gramáticas equivalentes em forma normal (reduzida) de Chomsky. Espera-se que a solução seja concisa, eficaz e elegante.

Problem

A sua tarefa é, dada um gramática livre de contexto G tal que $\epsilon \notin L(G)$, transformar G numa gramatica G' equivalente que se encontra na sua forma normal reduzida de Chomsky.

A forma normal reduzida de Chomsky e o algoritmo de transformação encontram-se definidos e detalhados em qualquer documento pedagógico sobre gramáticas livres de contexto (inclusive nos acetatos das aulas). Um resumo pode ser encontrado aqui: wikipedia/Forma_Normal_de_Chomsky

Ou no livro *An introduction to Formal Language and Automata (Fourth Edition)*, de Peter Linz, Jones and Bartlett Publishers, 2006. O algoritmo usado neste problema é exposto no teorema 6.6 deste livro e resumido de seguida.

Assume-se uma gramática simplificada $G = (N, T, S, P)$ tal que $\epsilon \notin L(G)$ e que G não possuí nem produções unitárias (da forma $A \to B$) nem produções epsilon (da forma $A \to \epsilon$).

Transformação 1: Construir uma gramática $G_1 = (N_1, T_1, S, P_1)$ considerando cada produção de P da forma $A \to x_1 x_2 \dots x_n$ (onde $x_i \in (N \cup T)$). Se $n = 1$ então $x_i \in T$ (por não haver produções unitárias), então incluir esta produção em P_1.

se $n > 1$ então considerar um novo simbolo não terminal V_a por cada terminal $a \in T$ ocorrendo na produção (sem duplicação). Substituir a por V_a nas produção em causa. Juntar finalmente a regra $V_a \to a$ em P_1 (novamente, sem duplicação).

No fim desta transformação, as regras de P_1 tem forçosamente uma das duas formas seguintes:

$$A \to a \quad (A \in N, a \in T)$$

$$A \to C_1 C_2 \dots C_n \quad (A, C_i \in N)$$

Transformação 2: Vamos transformar as regras da forma $A \to C_1 C_2 \dots C_n$. Basta fazer considerar, para esta configuração, as regras

$$A \to C_1 D_1$$
$$D_1 \to C_2 D_2$$
$$\dots$$
$$D_{n-2} \to C_{n-1} C_n$$

É possível realizar as transformações 1 e 2 em simultâneo desde que haja cuidado com as duplicações desnecessárias de regras e não terminais criados.

Input

A primeira linha apresenta o número m de regras de produção da gramática considerada.

As restantes m linhas introduzem as m regras de produção. Cada uma destas produções tem o formato seguinte `N -> a1 a2 a3 ... an` com N não terminal e $a_i \in (N \cup \Sigma)$ e cada a_i é separado do a_j seguinte por um espaço único.

Output

A gramática resultante é dada em saída, uma produção por linha e seguindo o mesmo formato da gramática em entrada.

Os não terminais novos gerados terão por nome V_i. O primeiro gerado será `V1`, o segundo `V2` etc... A ordem seguida corresponde a ordem das regras da gramática original e da leitura da esquerda para a direita.

As regras da gramática resultante são apresentadas seguindo a ordem $S, A, B, C \ldots$

Para duas regras com o mesmo *lhs*, segue-se a ordem lexicográfica sobre o *rhs*.

Constraints

Assume-se que o conjunto de não-terminais é $N = \{A, B, \ldots, S\}$ em que se destaca o simbolo S que assumiremos como sendo sempre o símbolo inicial. O alfabeto para os terminais é $\Sigma = \{a, \ldots, z\}$. O símbolo ♯ representa a palavra vazia ϵ.

A gramática em entrada não tem mais do que 50 regras e não tem regras de comprimento maior do que 10 símbolos gramaticais.

Sample Input 1

```
5
S -> A B
S -> a B
A -> a a b
A -> c
B -> b b A
```

Sample Output 1

```
S -> A B
S -> V1 B
A -> V1 V2
A -> c
B -> V3 V4
V1 -> a
V2 -> V1 V3
V3 -> b
V4 -> V3 A
```

Sample Input 2

```
4
S -> A b A A
S -> b c A
A -> S S b
A -> a
```

Sample Output 2

```
S -> A V1
S -> V2 V3
A -> S V4
A -> a
V1 -> V2 V5
V2 -> b
```

```
V3 -> V6 A
V4 -> S V2
V5 -> A A
V6 -> c
```