Push-Down Autometa 12th Project

Specifications

Input:

A sample **grammar.txt** file is given below. First line states the terminals (alphabet) of the language. Second line gives the Non-terminals of the grammar. Note that first non-terminal is the **start symbol.** After these initial two lines, rules of the grammars are given (one rule on each line). Each line starts with the ID of the rule (an integer). Every term in the grammar file is separated by a single space character. For simplicity, you can assume that all the terminals and non-terminals will be just a single character.

```
) ( + 0 1 * True False if x == y
S E I
1 S -> ( E )
2 S -> E + E
3 S -> E - E
4 S -> E * E
5 S -> E == E
6 E -> S
7 E -> 0
8 E -> 1
9 E -> I
10 I -> True
11 I -> False
12 E -> x
13 E -> y
14 E -> if
15 S -> E ( E )
16 S -> E > E
17 S -> E < E
```

We can add more grammar depends on what we want

Mohamed Hazem
18107076

The **words.txt** file will contain words (one word on each line). An example words.txt file is given below.

```
(0+1)+1        if(x==y)        True*False        if(x>y)
```

Output:

For each word in the **words.txt** file, if the word cannot be generated by the grammar you will only print NO. If the word is in the language, you should print YES in the first line and on the next line you will print the rules you applied to generate the word with a **left-most derivation**. The results should be printed to **output.txt** file. For the grammar and words file given in the previous section, your **output**.txt file should be as follows:

```
YES
2  6  1  6  2  7  8  8
YES
15  14  6  5  12  13
YES
4  |__IF Detected__||__Return False__|11
YES
15  14  6  16  12  13
```