Uniwersytet Jagielloński Instytut Informatyki KIS Katedra Informatyki i Stosowanej	Metody programowania 2021/2022 Konwersje: ONP ⇔ INF	P_03

Opis

Napisz efektywny program w Javie, który będzie realizował następujące operacje:

1. Konwertuje wyrażenia arytmetyczne i instrukcje przypisania z notacji INF do ONP.
2. Konwertuje wyrażenia arytmetyczne i instrukcje przypisania z ONP do notacji INF, zawierającej minimalną liczbę nawiasów, gwarantującą taką kolejność obliczeń jak w wyrażeniu ONP.

Wyrażenia arytmetyczne mogą zawierać jedynie:

a. nawiasy: (,) - tylko w notacji INF
b. operandy: małe litery alfabetu angielskiego
c. operatory, których priorytety i łączności przestawia tabela:

operator	priorytet	łączność	opis operatora
( )	najwyższy	lewostronna	nawiasy
!, ~		prawostronna	negacja , - unarny
^		prawostronna	potęgowania
*, /, %		lewostronna	multiplikatywny
+, -		lewostronna	addytywny
<, >		lewostronna	relacje < i >
?		lewostronna	relacja równości
&		lewostronna	koniunkcja
I		lewostronna	alternatywa
=	najniższy	prawostronna	przypisania

Podczas konwersji program usuwa znaki, które nie mogą występować w zadanych wyrażeniach, takie jak spacje, przecinki itp. oraz sprawdza poprawność składniową wyrażeń.

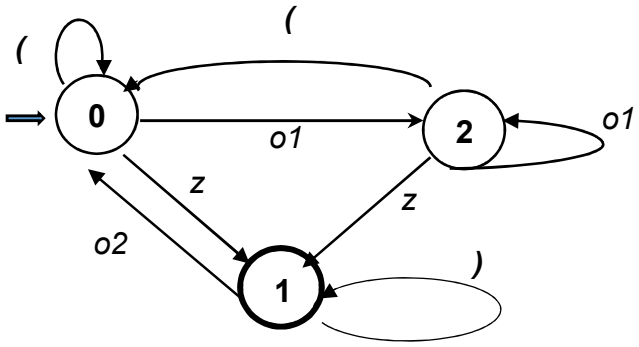
Przy czym można założyć, że po usunięciu zbędnych symboli: badana jest poprawność wyrażeń w postaci INF, opisana w następnym punkcie.

W przypadku wyrażenia w ONP, wyrażenie uważamy za poprawne jeśli jest wykonalne.

Poprawność wyrażeń arytmetycznych w postaci infiksowej (INF)

Badanie poprawności wyrażeń arytmetycznych składa się z dwóch kroków:

1. Sprawdzenie, czy wyrażenie jest akceptowane przez poniższy automat skończony:
   A=(Q, T, δ, 0, F), gdzie Q={0, 1, 2} – zbiór stanów, 0 - stan początkowy,
   F = {1} – zbiór stanów końcowych
   T = { z, o1, o2, (,) } – alfabet symboli, które mogą wystąpić w wyrażeniu,
   przy czym:
   z - operand - zmienna (pojedyncza litera),
   o1 - operatory jednoargumentowe { ~ , !)
   o2 - operatory dwuargumentowe { ^, *, /, %, +, -, <, >, ?, &, |, =}
   ( ) - nawiasy
   δ - funkcja przejścia automatu, którą definiuje poniższy graf



Automat rozpoczyna w stanie 0 analizę wyrażenia w INF od pierwszego symbolu wyrażenia. Jeśli automat zakończy analizę wyrażenia w INF znajdzie się w stanie 1 wówczas powiemy, że wyrażenie jest poprawne.
Przykłady błędnych wyrażeń: INF: a~+b , INF: a+b! , INF: ()a+b INF: (a+b)+() , INF: ~()a

2. Sprawdzenie, czy nie występują w wyrażeniu następujące przypadki:
   a. niesparowane nawiasy lub ich zła kolejność, np.) (a (
   b. niezgodność liczby operatorów i operandów, np. a+b*

Ze względu na efektywność analizy, kroki 1 i 2 powinny być wykonywane w pętli wykonującej konwersję wyrażenia.

Wejście

Dane do programu wczytywane są ze standardowego wejścia zgodnie z poniższą specyfikacją. Pierwsza linia wejścia zawiera liczbę całkowitą *z,* oznaczającą liczbę linii zawierających wyrażenia arytmetyczne lub instrukcje przypisania, których opisy występują kolejno po sobie.

Każda linia zawiera co najmniej 6 znaków i nie przekracza 256 znaków, może mieć jedną z dwóch postaci:

INF: wyrażenie arytmetyczne lub instrukcja przypisania, zapisane w notacji infiksowej,

ONP: wyrażenie arytmetyczne lub instrukcja przypisania, zapisane w notacji ONP.

Ostatnia linia każdego zestawu zakończona jest znakiem '\n'.

Wyjście

- Wyrażenie poprzedzone na wejściu napisem "`INF: `" musi być na wyjściu poprzedzone napisem "`ONP: `" i analogicznie wyrażenie poprzedzone na wejściu napisem "`ONP: `" musi być na wyjściu poprzedzone napisem "`INF: `". W przypadku błędnego wyrażenia, na wyjściu, zamiast skonwertowanego wyrażenia pojawi napis *error.*
- W przypadku konwersji wyrażenia w ONP do w INF, wyrażenie w INF musi zawierać minimalną liczbę nawiasów, gwarantującą podczas obliczania taką kolejność operacji (uwzględniając typ łączności i priorytety operatorów) jak w wyrażeniu ONP,
  np. ONP: x a b c * *= zostanie przekształcone do INF: x = a * (b * c)
- W trakcie konwersji program powinien usuwać znaki niewystępujące w wyrażeniach arytmetycznych lub w instrukcjach przypisania, w tym spacje oraz sprawdzać poprawność wyrażeń.
- W przypadku wyrażeń w notacji INF, np. INF: (a , + b) /. . [c 3 , program pozostawia jedynie: `(a+b)/c`, pozostałe znaki, w tym spacje – zignoruje i po sprawdzeniu poprawności wyrażenia, wypisze na wyjściu: `ONP: a b + c /`
- W przypadku wyrażeń w notacji `ONP`, np. `ONP: ( a,b,.).c;-,*` program pozostawia jedynie: `a b c - *` i po sprawdzeniu poprawności, dokona konwersji, wypisując na wyjściu: `INF: a * ( b - c )`
- Wszystkie elementy wyrażeń na wyjściu są poprzedzone pojedynczą spacją.

Wymagania implementacyjne

1. Ogólnie jak w poprzednich programach, w szczególności jedynym możliwym importem jest import skanera wczytywania z klawiatury. Tym samym klasę stosu należy zaimplementować samodzielnie.
2. Przypominam o komentowaniu aplikacji w formie opisanej w punkcie 3 Regulaminu zaliczania programów na BaCy.

Przykład danych

wejście:	wyjście:
12	
ONP: xabc**=	INF: x = a * (b * c)
ONP: ab+a~a-+	INF: a + b + (~ a - a)
INF: x=~~a+b*c	ONP: x a ~ ~ b c * + =
INF: t=~a<x<~b	ONP: t a ~ x < b ~ < =
INF: (a,+ b)/..[c3	ONP: a b + c /
ONP: (a,b,.).c;-,*	INF: a * (b - c)
ONP: abc++def++g+++	INF: error
INF: x=a=b=c^d^e	ONP: x a b c d e ^ ^ = = =
INF: (r+y)=a=(b+c)+d	ONP: r y + a b c + d + = =
INF: x=!(c>a & c<b)	ONP: x c a > c b < & ! =
INF: x=~~~a	ONP: x a ~ ~ ~ =
ONP: xa~~~=	INF: x = ~ ~ ~ a