Uniwersytet Jagielloński Instytut Informatyki i Matematyki Komputerowej KIS Katedra Informatyki Stosowan	Metody programowania 2015/2016 Konwersje: ONP ⟺ INF	P03

Opis

Napisz program w Javie, który będzie realizował następujące operacje:

1. Konwersja wyrażeń arytmetycznych z tradycyjnej notacji infiksowej do ONP
2. Konwersja wyrażeń z ONP do notacji infiksowej z **minimalną liczbą użytych nawiasów**.

Wyrażenia mogą zawierać jedynie:

a. nawiasy: (,) - tylko w notacji infiksowej
b. operandy: małe litery alfabetu angielskiego
c. operatory, wymienione poniżej w kolejności malejących priorytetów:
   - minus unarny: ~
   - potęgowanie: ^
   - operatory multiplikatywne: *, / , %
   - operatory addytywne: +, -
   - operatory relacji: <, >
   - operator negacji: !
   - operatory logiczne: & (and)
   - operatory logiczne: | (or)
   - operator przypisania: =

Wejście

Dane do programu wczytywane są ze standardowego wejścia (klawiatury) zgodnie z poniższą specyfikacją. Pierwsza linia wejścia zawiera liczbę całkowitą *z* – liczbę linii zawierających wyrażenia arytmetyczne, 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 zapisane w notacji infiksowej
<ONP> wyrażenie arytmetyczne zapisane w notacji ONP

Przy czym wyrażenia mogą zawierać dowolne znaki. Program najpierw usuwa znaki niewystępujące w wyrażeniach, w tym spacje a następnie sprawdza poprawność wyrażeń.

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 do notacji infiksowej, wyjściowe wyrażenie musi zawierać minimalną liczbę nawiasów gwarantującą taką kolejność operacji, jak w wejściowym wyrażeniu, np. <ONP>abc** zostanie przekształcone do <INF>a*(b*c) a nie do

<INF>a*b*c. Nawiasy obejmujące b*c w wyrażeniu wyjściowym wymuszają taką kolejność operacji mnożenia jaka jest w zapisie ONP w wyrażeniu wejściowym.

- W przypadku wyrażeń w postaci infiksowej, np. <INF> (a,+ b)/..[c3 , program pozostawia jedynie: (a+b)/c, pozostałe znaki, w tym spacje – wyrzuca, dodatkowo sprawdza poprawność wyrażenia, po czym dokonuje konwersji, wypisując na wyjściu: <ONP>ab+c/ .
- W przypadku wyrażeń w notacji ONP, np. <ONP>(a,b,.) .c;-,* program pozostawia jedynie: abc-*, dodatkowo sprawdza, czy wyrażenie jest poprawne, po czym dokonuje konwersji, wypisując na wyjściu: <INF>a*(b-c).
- Wyrażenia w notacji infiksowej postaci <INF> a<x<b uważamy za poprawne.

Wymagania implementacyjne

Ogólnie jak w poprzednich programach, w szczególności jedynym możliwym importem jest import skanera wczytywania z klawiatury. Tym samym klasę stosu należy zaimplementować samodzielnie.

Przykład danych

wejście:	wyjście:
16	
<ONP> a,b,. ^ .c;-,*	<INF>error
<INF> ((a,+ b)/..[c3	<ONP>error
<INF>(a+b)*c+(d-a)*(f-b)	<ONP>ab+c*da-fb-*+
<INF>~~~a	<ONP>error
<ONP>ab+c*da-fb-*+	<INF>(a+b)*c+(d-a)*(f-b)
<ONP>ab*c*d*e*	<INF>a*b*c*d*e
<ONP>(a # b .c*))*	<INF>a*(b*c)
<ONP>ab<c/d+	<INF>(a<b)/c+d
<INF>~ a, ($ /(b*c^d)	<ONP>error
<ONP>a~~~	<INF>~(~(~a))
<ONP>zab+*>!	<INF>error
<INF>~a*(~b+~c)/~(~d-e)	<ONP>a~b~c~+*d~e-~/
<ONP>ab~c+*de-~/	<INF>a*(~b+c)/~(d-e)
<INF>)a+b(	<ONP>error
<INF>,z,a<x.>b!=	<ONP>error
<ONP>a*aa+	<INF>error