ESERCIZI SU ANALISI LESSICALE

3/2/2023

Si consideri la seguente grammatica (scritta in ANTLR)

```
prg :  'let' dec 'in' stm ;
dec :  ('int' Id ';')+ ;
exp :  Integers | Id | exp '+' exp ;
stm :  (Id '=' exp ';')+
```

dove

- gli `Integers` sono sequenze non vuote di cifre prefissate dal segno `+` o `-`;
- gli `Id` sono gli identificatori (sequenze non vuote di caratteri);

Esercizi

1. (**punti 2**) completare l'input di ANTLR con le regole per l'analizzatore lessicale che riguardano `Integers` e `Id`;

16 Settembre 2022

Esercizio 1 (6 punti) Si definisca un analizzatore lessicale in `ANTLR` che accetta sequenze di token che a loro volta sono stringhe (non vuote) sull'alfabeto a, b che contengono un numero pari di occorrenze di b.

20 Dic2mbre 2021

Esercizio 1 (6 punti). Sia L il linguaggio sull'alfabeto $\{a, b, c, d\}$ costituito da sequenze (non vuote) di token della forma $\alpha d \beta$ dove α è una qualunque stringa non vuota che contiene $\{b, c\}$ e β è una qualunque stringa non vuota che contiene $\{a, c\}$. Ad esempio `ccdc bdc` è una sequenza di token valida, mentre `cc ada` è sbagliata. Si definisca in `ANTLR` l'analizzatore lessicale per tokens in L senza utilizzare gli operatori `*` o `+`.

18/9/2020

Esercizio 1 (punti 6) Gli identificatori di un linguaggio di programmazione devono iniziare e terminare con "_" e tra questi due caratteri ci possono essere solo lettere maiuscole e cifre (in qualunque ordine) con il vincolo che il numero di lettere e quello delle cifre sia sempre pari. Definire l'analizzatore lessicale per questi identificatori in `ANTLR`.

15/6/2020

Esercizio 1 (6 punti). Definire un analizzatore lessicale in `ANTLR` che accetta sequenze di token che a loro volta sono stringhe non vuote sull?alfabeto {`a`, `b`} per cui non ci sono mai due occorrenze di `b` consecutive. Ad esempio `a abaa b aaaab` è un input riconosciuto.

19 Settembre 2019

Esercizio 1 (6 punti). Definire un analizzatore lessicale in `ANTLR` che accetta sequenze di token che a loro volta sono stringhe **non vuote** sull'alfabeto a, b, c, per cui le occorrenze di a (se ci sono) precedono le occorrenze di b e di c (se ci sono) e le occorrenze di b precedono quelle di c (se ci sono). Ad esempio `a abbc bcc c` è un input riconosciuto.

4 Luglio 2019

Esercizio 1 (6 punti). Dato l'input di ANTLR

```
start : (BINDIGIT PIU DIGIT)+ ;
PIU : '+' ;
BINDIGIT : ('0' | '1')+  ;
DIGIT : ('0'..'9')+  ;
WS     : (' ' | '\t' | '\n' | '\r' ) -> skip ;
```

Dire cosa accade quando l'input da analizzare è (motivare le risposte):

a) `1+1`

b) `1+2+3`

5 Giugno 2019

Esercizio 1. Definire un analizzatore lessicale in `ANTLR` che accetta sequenze di token che a loro volta sono stringhe sull'alfabeto a, b, c, che contengono esattamente una e una sola occorrenza di a ed una e una sola occorrenza di b.