Progettino LPC A.A. 2008/2009
(ASCOLI PICENO)
Ver. 2.1

Sia definito il seguente linguaggio AP^{2}SL utilizzato per definire testualmente **automi a pila deterministici** che accettano per pila vuota (\n rappresenta la nuova riga o il ritorno carrello che in alcuni sistemi è rappresentata invece da \r, mentre $ rappresenta il simbolo di pila vuota e # rappresenta il simbolo epsilon):

AP2SL ::= "Automa :" <NOME>"\n"{<STATI><COPPIA>"\n"<TABTR>"."
COPPIA::= ","<STATI><COPPIA>"\n"<TABTR>":" | "}\n{"<LISTAALFABETO>"}\n{"<LISTASIMBOLIPILA"}"
LISTASIMBOLIPILA ::= "$"(","<SPILA>)+
LISTAALFABETO ::= <ALFABETO>(","<ALFABETO>)*
TABTR ::= "{"<TRANSIZIONI>"}"
TRANSIZIONI ::= "("(<SPILA>|"$")","(<ALFABETO>|"#")")->("<STATI>","<APILA>");"<TRANSIZIONI> |
"("(<SPILA>|"$")","(<ALFABETO>|"#")")->("<STATI>","<APILA>")"
NOME ::= ["A" - "Z"](["a" - "z", "0" - "9", "_"])+
STATI ::= ["1"-"9"](["0"-"9"])*
ALFABETO ::= ["a" - "z"]
SPILA ::= ["A" - "Z"]
APILA ::= (<SPILA>)*

Il non terminale COPPIA è necessario per far si che non sia possibile definire un numero di linee generate da "TABTR" differenti dal numero di elementi "STATI" dunque per ogni stato dovrà essere definito un insieme di transizioni eventualmente vuoto. In pratica ciò corrisponde ad una situazione del tipo a^nb^n, caratterizzabile tramite un linguaggio libero da contesto.

In pratica il linguaggio permette di definire un automa a pila deterministico che accetta per pila vuota. L'assunzione è che il primo stato elencato sia lo stato iniziale e che l'elenco di insiemi di transizioni rispetti lo stesso ordine utilizzato per la descrizione dell'insieme degli stati. A titolo di esempio la seguente definizione di automa corrisponde all'automa che permette di accettare il linguaggio a^nb^n scritto nel linguaggio AP^{2}SL:

```
Automa : Automa_anbn
{1,2}
{a,b}
{$,N}
{($,a)->(1,N);(N,a)->(1,NN);(N,b)->(2, )}
{(N,b)->(2, )}.
```

Lo studente dovrà definire un file .jj che gli permetta di derivare un compilatore per programmi scritti in AP^{2}SL. In particolare un programma AP^{2}SL una volta in esecuzione dovrà essere capace di prendere in ingresso una stringa di caratteri e stampare a video tutti i passi effettuati dal corrispondente automa a pila nell'accettazione o non accettazione della stringa di ingresso.
Il seguente potrebbe/dovrebbe rappresentare l'output a video del programma java ottenuto dalla compilazione per l'automa di cui sopra, sulla stringa "aabb":

```
Automa: Automa_anbn
Alfabeto: a,b
Simboli di Pila: N

Inserisci la stringa di input: aabb
STATO: 1 PILA: $ input: aabb  - AZIONE: 1,N
STATO: 1 PILA: N$ input: abb  - AZIONE: 1,NN
STATO: 1 PILA: NN$ input: bb  - AZIONE: 2,
STATO: 2 PILA: N$ input: b  - AZIONE: 2,
STATO: 2 PILA: $ input:  - AZIONE: ACCETTO
```

Ci sono una serie di situazioni di errore nella definizione di un programma AP^{2}SL che non possono essere espresse e riconosciute direttamente attraverso la corrispondente grammatica libera da contesto. Tali condizione di errore potrebbero invece essere riconosciute dal compilatore aggiungendo del codice opportuno. Una possibile lista di situazioni di errore da poter notificare al programmatore sono:

1. La lista dei caratteri dell'alfabeto potrebbe contenere duplicati,
2. La lista dei simboli di pila potrebbe contenere duplicati
3. La lista degli stati potrebbe contenere duplicati
4. La transizione incontrata potrebbe essere già stata definita per quella configurazione
5. Il primo insieme di transizioni (corrispondente alle transizioni per lo stato iniziale) non contiene nessuna transizione a partire dalla pila vuota
6. Lo stato definito per una transizione non è tra gli stati dell'automa
7. Il carattere di input specificato in una transizione non fa parte dell'alfabeto
8. I caratteri di Pila specificati in una transizione non fanno parte dell'alfabeto di pila

Dunque in definitiva i passi del progetto dovrebbero essere i seguenti:

1. definizione di un file .jj che riporta la grammatica del linguaggio AP^{2}SL. Attraverso JavaCC lo studente potrà derivare una classe `ap2slc.java` che una volta compilata restituirà un compilatore in java per il linguaggio AP^{2}SL.
2. a quel punto potranno essere definiti file generici `nome.aps` ognuno dei quali rappresenterà la definizione di un automa a pila.
3. i file .aps potranno essere compilati con il comando `java ap2slc nome.aps` ottenendo un file `nome.java` che rappresenta il corrispondente automa scritto nel linguaggio java. Una volta compilato con il comando `javac` l'automa potrà essere lanciato con il comando `java nome`
4. Una volta lanciato l'automa a pila sarà possibile passare una stringa per verificarne o meno l'appartenenza al linguaggio accettato dall'automa.

Suggerimenti:
Si ricorda che la funzione di transizione di un automa a pila deterministico può essere facilmente rappresentata da una matrice tridimenzionale. Nel caso del linguaggio definito la matrice dovrebbe essere riempita via via che le azioni vengono identificate dal parser.
Ricordo poi che in java è possibile definire nuovi tipi attraverso il costruttore di tipi enumerazione. Potrebbe esser comodo ricondurre l'insieme delle etichette e degli stati ad enumerazioni con numeri progressivi da poter dunque utilizzare come indici di matrici.

Per ogni dubbio o incertezza sulla specifica potete contattarmi al mio indirizzo e-mail andrea.polini@unicam.it

Versioni
V2.0 → V2.1

E' stato modificato un errore nell'esempio dove il simbolo epsilon (#) veniva utilizzato per rappresentare il fatto che non venivano inseriti elementi nella pila. Tale fatto è invece rappresentato dalla grammatica dalla mancanza di elementi tra la virgola e la parantesi tonda nella parte sinistra di una produzione.