

PROGRAMMAZIONE – V Appello del 9/06/2003

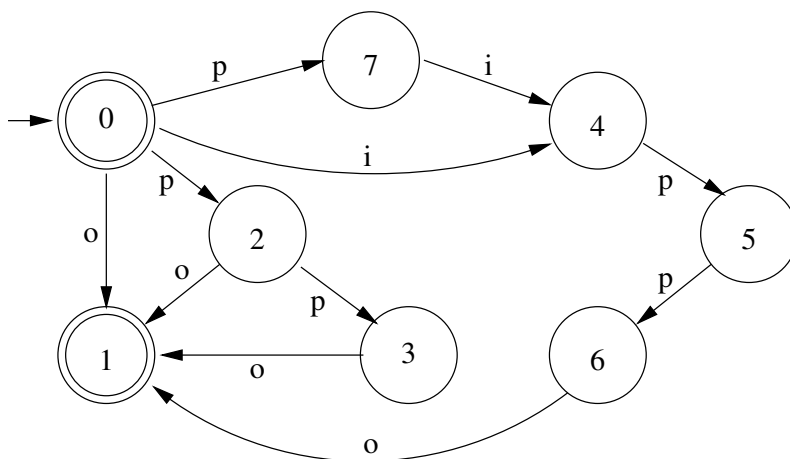
Scrivere **in stampatello** COGNOME, NOME e NUMERO DI MATRICOLA (se conosciuto) su ogni foglio consegnato e sul testo, che va consegnato insieme al compito.

ESERCIZIO 1 (6 punti)

(1) Disegnare un automa sull'alfabeto dei caratteri ASCII che riconosca tutti e soli i suffissi della parola **pippo** (anche ϵ e **pippo** devono essere considerati suffissi).

(2) Costruire una grammatica *regolare* che genera lo stesso linguaggio.

SOLUZIONE



```
<0> ::= epsilon | p<7> | p<2> | o<1> | o | i<4>
<1> ::= epsilon
<2> ::= o<1> | o | p<3>
<3> ::= o<1> | o
<4> ::= p<5>
<5> ::= p<6>
<6> ::= o<1> | o
<7> ::= i<4>
```

ESERCIZIO 2 (12 punti)

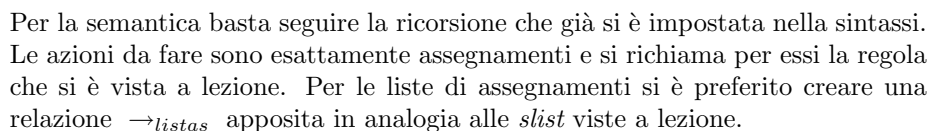
(1) Si scrivano le produzioni per una categoria sintattica **ListAs** che generi sequenze di assegnamenti lunghe almeno due in cui gli assegnamenti sono separati da $->$. Per generare le espressioni aritmetiche o booleane si usi la sottocategoria sintattica **Exp** vista a lezione e per generare gli identificatori si usi la sottocategoria sintattica **Ide** vista a lezione. Ad esempio la seguente è una sequenza di assegnamenti:

```
pippo = 3 + x; -> pluto = pippo * (3 * pluto); -> pippo = x + 1;
```

(2) Si scrivano poi le regole di semantica operativa per le liste di assegnamenti generate da **ListAs**. L'esecuzione di una lista di assegnamenti deve portare in uno stato in cui si sono eseguiti in sequenza tutti gli assegnamenti della lista a partire dal primo a sinistra. Si faccia riferimento allo stato formato solo da una pila di frame.

La grammatica ha una semplice ricorsione a destra e un caso base in cui ci sono due assegnamenti. Si noti che la ricorsione a destra servirà poi anche per poter scrivere in maniera semplice le regole per la semantica data.

Il seguente è l'albero di derivazione per la lista data. Per gli identificatori e le espressioni non si sono disegnati gli alberi, ma si è solo indicato quale deve essere la frontiera.



2

ESERCIZIO 3 (6 punti)

Si scriva un metodo `cerca13` che prenda un array `a` di interi come parametro e che restituisca il numero di occorrenze del sottoarray `[1|3]` nell'array `a`.

Esempio: `a = [0|5|15|1|3|23|1|2|3|0|150|2|1|3|12]`. Il metodo deve restituire l'intero 2 in quanto ci sono 2 occorrenze dell'array `[1, 3]`, una in posizione 3 e l'altra in posizione 12 dell'array `a`.

SOLUZIONE

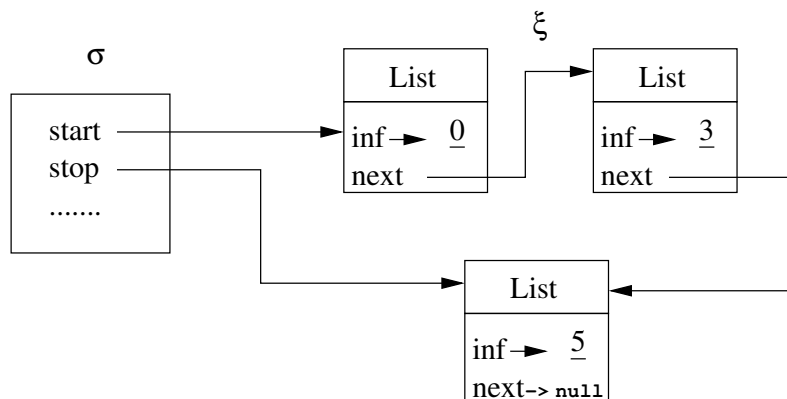
```
public int cerca13 (int [] a) {  
    int conta = 0;  
    // arrivo a controllare fino alla penultima posizione dell'array  
    // poiche' [1,3] non puo' cominciare nell'ultima posizione  
    for (int i = 0; i < a.length - 1; i++)  
        if (a[i] == 1 && a[i+1] == 3)  
            conta = conta + 1;  
    return conta;  
}
```

ESERCIZIO 4 (6 punti)

Si consideri la seguente definizione di classe:

```
class List {  
    public int inf;  
    public List next;  
  
    public void reset() {  
        this.inf = 0;  
        this.next = null;  
    }  
}
```

Si scriva un programma che utilizzi la classe `List` e che termini in uno stato contentente almeno gli oggetti e le variabili rappresentati in figura.



SOLUZIONE

```
{  
    List start = new List;  
    start.inf = 0;  
    List app = new List;  
    app.inf = 3;  
    start.next = app;  
    List stop = new List;  
    app.next= stop;  
    stop.inf = 5;  
    stop.next = null;  
}
```