Corso di Linguaggi di Programmazione — Paradigmi di Programmazione
Prova scritta del 10 febbraio 2023.

Tempo a disposizione: ore 2.
Svolgere gli esercizi 1–4, 5–6 e 7–8 su due fogli differenti.

1. Descrivere le regole di semantica operazionale strutturata per l'espressione booleana b_0 `or` b_1, secondo la disciplina di valutazione interna-parallela (IP). Mostrare un esempio di una espressione di quel tipo tale che la valutazione IP e quella ES (esterna-sinistra, vista a lezione) non sono uguali.

2. Determinare tutti gli item LR(0) che si possono ottenere dalla grammatica $S \to \epsilon \mid \mathtt{a}S\mathtt{b}$. Quale linguaggio genera questa grammatica?

3. Si consideri la grammatica G con simbolo iniziale S:

$$\begin{array}{lcl} S & \to & SAB \mid \epsilon \\ A & \to & a \mid aA \\ B & \to & b \mid bB \end{array}$$

   (*i*) Quale linguaggio genera? (*ii*) Manipolare la grammatica G, rimuovendo la ricorsione sinistra immediata su S. (*iii*) Quindi fattorizzare le produzioni per A e B. (*iv*) Verificare se la risultante grammatica sia di classe LL(1).

4. Si consideri il seguente NFA $M = (\Sigma, Q, \delta, q_0, F)$, dove $\Sigma = \{a\}$, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $F = \{q_2, q_3\}$ e la funzione di transizione $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ è cosí definita: $\delta(q_0, a) = \{q_1, q_2\}$, $\delta(q_1, a) = \{q_4\}$, $\delta(q_2, a) = \emptyset$, $\delta(q_3, a) = \emptyset$, $\delta(q_4, a) = \{q_1, q_5\}$, $\delta(q_5, a) = \emptyset$, $\delta(q_1, \epsilon) = \{q_3\}$ e, infine, $\delta(q, \epsilon) = \emptyset$ per $q \in \{q_0, q_2, q_3, q_4, q_5\}$.

   (*i*) Si fornisca una rappresentazione grafica di M. (*ii*) Si determini il DFA M' minimo associato a M. (*iii*) Qual è il linguaggio riconosciuto da M'? (*iv*) Fornire una espressione regolare che rappresenti tale linguaggio.

5. Si consideri l'implementazione dello scope statico mediante display. Si dica, motivando la risposta, se dimensione massima del display può essere determinata durante la compilazione.

6. Si dica cosa stampa il seguente frammento di programma, in uno pseudolinguaggio con scope statico e passaggio dei parametri per nome.

```
int i = 1;
int[] A = new int[5];
void fie (int x, int y) {
        int i = 3
        x = x+1;
        write(y);
        y = 1;
        write(A([i]));
        A([i]) = 77 ;
        }
for (j = 0; j <= 4; j+= 1)
        {A[j] = 0};

fie (i,A[i]);
write([A(1)]);
write([A(2)]);
write([A(3)]);
write([A(i)]);
```

7. Si consideri un linguaggio con passaggio per valore nel quale le eccezioni sono dichiarate con la sintassi `exception E` (`E` nome dell'eccezione), sono sollevate con l'istruzione `throw E` e sono gestite coi blocchi `try { ... } catch E { ... }`. Il linguaggio ha scoping statico per tutti i nomi, eccezioni comprese. Cosa stampa (tramite l'operazione `print`) il seguente frammento? Spiegare brevemente il ragionamento dietro la risposta.

```
exception Y;
a() { throw Y; }
exception X;
b( int x ) {
 if ( x < 5 ){ print( x ); a(); }
 else { print( x ); throw X; }
}
c( int x ) {
 try { b( x++ ); }
 catch ( Y y ) { print( 1 ); c( x ); }
}
d( int z ) { try { c( z ); } catch ( X x ){} }

d( 3 );
```

8. Il pseudolinguaggio usato nel codice sottostante ammette l'uso dei puntatori—`new A()` alloca una nuova struttura di tipo `A` nello heap e `free( a )` libera la memoria nello heap puntata da `a`. Il codice presenta problemi nella gestione dei riferimenti. Dove? In che modo la tecnica delle "tombstones" risolverebbe il problema? Motivare la risposta.

```
struct A { struct A* a; int b; };
A* p;
A* a = new A();
*a.a = a;
p = a;
free( p );
*a.b = 5;
```