Corso di Linguaggi di Programmazione
Prova scritta del 4 Febbraio 2025.

Tempo a disposizione: ore 2.
Svolgere gli esercizi 1–4, 5–6 e 7–8 su tre fogli separati.
Scrivere nome, cognome e matricola su ogni foglio consegnato.

FOGLIO 1 ▷ 1. La seguente espressione $\mathcal{I}_{L_2}^{L_1}(\mathcal{C}_{L_2,L_1}^{L_2},\mathcal{C}_{L_2,L_1}^{L_2})$ cosa sta ad indicare? E cosa produce?

FOGLIO 1 ▷ 2. Si definisca il DFA minimo per il linguaggio $L=\{a^n \mid n=4\times k, k\geq 0\}$.

FOGLIO 1 ▷ 3. Si consideri la grammatica G con simbolo iniziale S:

$$\begin{array}{lcl} S & \rightarrow & BA \\ A & \rightarrow & \mathtt{b}A \mid \epsilon \\ B & \rightarrow & \mathtt{a}B\mathtt{b} \mid \epsilon \end{array}$$

(i) Quale linguaggio genera la grammatica G? (ii) Calcolare i first e i follow per tutti i nonterminali di G. (iii) Verificare se G sia di classe LL(1). (iv) In caso affermativo, costruire la tabella di parsing LL(1); altrimenti, argomentare se G possa essere di classe LL(k) per qualche k ≥ 2.

FOGLIO 1 ▷ 4. Si consideri la grammatica G con simbolo iniziale S del punto precedente. (i) Si costruisca la tabella di parsing SLR(1). (ii) Se non sono presenti conflitti, si mostri il funzionamento del parser SLR(1) per input *abbb*.

FOGLIO 2 ▷ 5. Si assuma che in un generico linguaggio imperativo a blocchi, senza ricorsione, durante l'esecuzione del blocco A venga invocata la funzione f. Il numero dei record di attivazione (RdA) presenti a run-time sulla pila fra il RdA di A e quello della chiamata di f è fissato staticamente o può variare dinamicamente? Motivare la risposta.

FOGLIO 2 ▷ 6. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e passaggio di parametri per nome e per valore.

```
int x = 50;
void pippo(name int y, value int z){
     z = y + y + x;
     }
   { int x = 1;
     int y = 10;
     int z = 20;
     pippo(++x, x);
     write(x);
     pippo(x++, x);
     write(x);
     }
write(x);
```

FOGLIO 3 ▷ 7. Si assuma di avere uno pseudolinguaggio che adotta la tecnica *locks and keys*. Dato `OGG` generico oggetto nello heap, indichiamo con `OGG.lock` il suo lock (nascosto). Data `p` variabile contenente il valore di un generico puntatore (sulla pila o nello heap), indichiamo con `p.key` la sua chiave (nascosta). Si consideri il seguente frammento di codice:

```
type C { C next }      // struttura del tipo C
C p = new C();         // OGG1
C q = new C();         // OGG2
for( int i = 0, i < 2, i++ ){
 p.next = q;
 q = p;
 p = p.next;
}
```

Si diano possibili valori di `OGG1.lock`, `OGG2.lock`, `p.key`, `p.next.key`, `q.key`, `q.next.key` dopo l'esecuzione del frammento (spiegare brevemente il ragionamento seguito).

FOGLIO 3 ▷ 8. Si rappresenti e descriva la struttura delle *vtable* corrispondenti alle seguenti dichiarazioni di classi, assumendo ereditarietà singola e che **class** B **extends** A indica un rapporto di ereditarità dalla classe B verso la classe A.

```
class A {
 int x = 1;
 int h( int n ){ return n + 1; }
}
class B extends A {
 int x = 1;
 int g(){ return f( 3 ); }
}
```

Dopo aver compilato le classi A e B, supponiamo di modificare la classe A come segue

```
class A {
 int x = 2;
 int h( int n ){ return x + 1; }
}
```

possiamo ricompilare solo A e usare con sicurezza il compilato ottenuto in precedenza della classe B? Motivare la risposta.