Corso di Linguaggi di Programmazione
Prova scritta del 4 Giugno 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. Fornire le regole di semantica operazionale per il costrutto b_0 **or** b_1, secondo la disciplina di valutazione interna-sinistra (IS).

FOGLIO 1 ▷ 2. Si definisca un semplice automa che generi il linguaggio $L = \{a^n b^m \mid n+m = 2\times k+1, k \geq 0\}$. Tale linguaggio è regolare? Motivare la risposta.

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

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

(i) Quale linguaggio genera la grammatica G? (ii) Si può costruire un DFA che riconosca G? (iii) Calcolare i first e i follow per tutti i nonterminali di G.

FOGLIO 1 ▷ 4. Si consideri la grammatica G con simbolo iniziale S del punto precedente. (i) G è ambigua? (ii) Si calcolino tutti gli item LR(0). (iii) Verificare se G sia di classe SLR(1).

FOGLIO 2 ▷ 5. Si consideri il seguente frammento di codice:

```
void power (int x, y, z){
   z = 1;
   while y > 0 do
    {z = z*x;
     y = y-1
     }
}
```

Assumendo che i parametri attuali `a` e `c` siano variabili che denotano la stessa locazione di memoria si elenchino tutte le combinazioni di modalità di passaggio dei parametri che possono essere usate affinché la chiamata `power(a,a,c)` lasci in `c` il valore $c{=}a^a$.

FOGLIO 2 ▷ 6. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usi scoping dinamico e deep binding:

```
int x = 3;
procedure stampa_x(){
    write_integer(x);
    }
procedure ass_x(n:int){
    x = n;
    stampa_x();
    }
procedure pippo(function S, P; int n){
    int x = 10;
    if (n=1) then  {ass_x(n);
                    stampa_x();
                   }
             else {S(n);
                   P();
                  }
    }
{
int x = 100;
pippo(ass_x, stampa_x, 1);
pippo(ass_x, stampa_x, 2);
}
```

FOGLIO 3 ▷ 7. Viene dato un linguaggio con tipaggio nominale implicito (indicato dal simbolo ?) e polimorfismo di sottotipo e parametrico. Sono dati i tipi `A`, `B` e `C` tali che `B <: A` e `C <: A`, cioè `B` e `C` sono sottotipi di `A`. `List[ T ]` supporta le operazioni `add: T -> ()` e `get: () -> T`, con `T` parametro di tipo. Le annotazioni dei parametri di tipo `? <: T` e `? :> T` indicano rispettivamente l'uso covariante e controvariante rispetto a `T`. Indicare quali istruzioni nel codice di `f` sono corrette e scorrette per il controllore di tipi. Per l'inferenza dei tipi impliciti dei parametri di `f` si applichi l'algoritmo di unificazione alle equazioni riportate a sinistra di `f` dove `X`, `Y` e `Z` corrispondono ai parametri con lo stesso nome (in minuscolo) di `f`. Spiegare in breve il ragionamento seguito.

```
K -> W = C -> X -> Y
C -> K = Z -> C
B -> W = Y -> A -> B
```

```
f (? x, ? y, ? z, List[? :> A] aa, List[? <: B] bb){
  x = y ;            /* I1 */
  x = bb.get();      /* I2 */
  aa.add( x );       /* I3 */
  bb.add( y );       /* I4 */
  aa = bb;           /* I5 */
  x = aa.get();      /* I6 */ }
```

FOGLIO 3 ▷ 8. Un linguaggio supporta la creazione di oggetti tramite la clausola **`object`**, specificando un nome di variabile seguito dalla definizione di metodi e campi (come un record). Il linguaggio supporta delegazione tramite la dichiarazione `X` **`prototypeOf`** `Y`, interpretato come "l'oggetto X è prototipo dell'oggetto Y". Indicare cosa stampa (**`print`**) il codice sotto, spiegando brevemente il ragionamento seguito. Argomentare se è possibile implementare lo stesso comportamento tramite ereditarietà tra classi in un linguaggio compilato, ad esempio, considerando la compilazione delle classi in momenti diversi.

```
object A {
  m(){ this.x = this.x * 2 }
  g(){ return this.x }
}
object D { m(){ this.x = this.x * 4 } }
object B { x = 1 }
A prototypeOf B
object C { x = 2 }
B prototypeOf C
C.m()
print( C.g() )
D prototypeOf B
C.m()
A prototypeOf B
print( C.g() )
```