Corso di Linguaggi di Programmazione
Prova scritta del 13 Gennaio 2026.

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. Siano date due classi di linguaggi formali A e B, tali che $A \subseteq B$. Si chiede se le seguenti due affermazioni siano vere. Giustificare la risposta. (Suggerimento: pensate alle classi dei linguaggi regolari, o liberi deterministici, o liberi.)

   (a) Se A è chiusa rispetto all'operazione di unione, allora anche B lo è.

   (b) Se B è chiusa rispetto all'operazione di unione, allora anche A lo è.

FOGLIO 1 ▷ 2. Il linguaggio $L = \{a^n b^m c^k \mid n,m,k \geq 0\}$ è regolare? Giustificare la risposta.

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

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

   (i) Quale linguaggio genera G? (ii) G è ambigua? In caso affermativo, manipolarla per renderla non ambigua. (iii) Manipolare G per ottenerne una equivalente senza produzioni unitarie.

FOGLIO 1 ▷ 4. Costruire un parser bottom-up per il linguaggio generato dalla grammatica G del punto precedente.

FOGLIO 2 ▷ 5. Si consideri il seguente frammento di codice assumendo che si abbia scope dinamico. Si dica cosa viene stampato in caso di deep binding e in caso di shallow binding.

```
var x := 1;

procedure P(f) {
    var x := 10;

    procedure R() {
        var x := 20;
        f();
        print(x);
    }

    R();
    print(x);
}

procedure Q() {
    print(x);
}

procedure S() {
    var x := 100;
    P(Q);
    print(x);
}

S();
print(x);
```

FOGLIO 2 ▷ 6. Il solito LLM di turno, a una domanda sulla differenza fra shallow biding e deep binding in caso di scope statico, risponde come segue. "Frase perfetta da esame: con scope statico il problema del deep e shallow binding non si pone, poiché le variabili non locali vengono risolte in base alla struttura lessicale del programma e l'ambiente è determinato al momento della definizione della funzione". E' davvero una "frase perfetta da esame" o si tratta di un'allucinazione? Commentare brevemente. (Suggerimento: *non* considerare l'esempio dell'esercizio precedente).

FOGLIO 3 ▷ 7. Uno pseudolinguaggio permette il dereferenziamento dei puntatori con l'operatore `*`, usato anche per indicare il tipo puntatore `A*`, `new A()` alloca una nuova struttura di tipo `A` nello heap, riportandone il puntatore e `free( a )` libera la memoria rispetto a un puntatore. Il codice seguente presenta problemi nella gestione dei riferimenti? Di che tipo? La tecnica "lock and keys" quali di questi problemi risolverebbe? Motivare la risposta.

```
struct A { int x; A* next; };
A* q;
A* r = new A();
*r.x = 10;
q = r;
*r.next = new A();
*r.next.x = 20;
free( r );
*q.next.x = 30;
free( q );
```

FOGLIO 3 ▷ 8. Assumendo uno pseudolinguaggio con tipi somma e prodotto e polimorfismo di sottotipo e universale, caratterizzare la definizione dei tipi `Maybe<T>` e `Result<T,E>` – si può usare la sintassi di definizione dei tipi `type A: ...` con `+` per somma, `x` per prodotto, `A <: B` per indicare che `A` è un sottotipo di `B` e `<A,B,...>` per i parametri di tipo. Il pseudolinguaggio supporta la definizione di operazioni con la sintassi `a( Input ): Ritorno`. Considerando le definizioni di `f`, `g` e `h` riportate sotto, indicare se l'istruzione `h(f(g('a')))` viene valutata come corretta per un controllore dei tipi, introducendo, nel caso servano, i necessari vincoli di sottotipaggio. Infine, indicare uno o più costrutti linguistici necessari per permettere l'uso dei tipi somma nelle implementazioni.

```
f( Maybe< Int > ) : Result< Char, NaN >

g( Char ) : Maybe< Int >

h( Result< Char, Error > ) : Char
```