Corso di Linguaggi di Programmazione
Prova scritta del 27 Maggio 2026.

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

FOGLIO 1 ▷ 1. Si dica cosa stampa il seguente frammento in uno pseudolinguaggio con passaggio per nome e scope dinamico.

```
int x = 5;

void foo(int y, int z) {
    x = x + 2;
    y = z + Z;
    write(x);
}
```

```
{
    int x = 20;
    foo(x, x++);
    write(x);
}
```

FOGLIO 1 ▷ 2. Il linguaggio imperativo MinRic contiene: assegnamenti, comandi di sequenza, selezione e iterazione, funzioni, allocazione esplicita della memoria e deallocazione esplicita della memoria.

Le funzioni ricorsive sono ammesse solo se ricorsive in coda.

Si dica, motivando la risposta, quale sia la forma più semplice di gestione della memoria utilizzabile per implementare MinRic.

FOGLIO 1 ▷ 3. Si consideri il seguente frammento scritto nello pseudolinguaggio Cfinto:

```
struct Cella {
    int val;
    struct Cella* next;
};

typedef Cella* Lista;
```

```
Lista p;
p = creaLista();

while ((p != NULL) && ((*p).val != 7)) do
    p = (*p).next;
```

Si supponga che `creaLista()` restituisca un puntatore a una lista.

Con una certa implementazione di Cfinto l'esecuzione termina senza errori; con un'altra implementazione corretta produce invece un errore a tempo di esecuzione.

Si fornisca una possibile spiegazione.

FOGLIO 1 ▷ 4. Si consideri il seguente frammento in uno pseudolinguaggio con parametri di ordine superiore:

```
{
    int a = 10; int b = 10;
    int g() {
        write(a, b);
    }
    void foo(int f(), int b) {
        int a = 30;
        if (b == 0) {
            f(); g();
        } else {
            a = 40; b = 40; foo(f, 0);
        }
    }
    {
        int a = 50; int b = 50; foo(g, 1); g();
    }
}
```

Si dica cosa stampa il frammento assumendo scope statico e deep binding.

FOGLIO 2 ▷ 5. Uno pseudolinguaggio supporta i tipi `f: AxB -> C` dove la funzione `f` ha `A` e `B` tipi dei parametri e `C` tipo di ritorno, `A(X,Y) = ...` polimorfo universale con `X`, `Y` parametri di tipo e `A + B` somma. Si vuole costruire una piccola pipeline tramite tre funzioni: `input` ritorna una stringa da input dell'utente e può fallire con un errore di tipo `IOError`, `parse` effettua il parsing da stringa a intero e può fallire con un errore di tipo `ParseError`, `div` calcola la divisione intera di 42 per un intero dato in input e può fallire con un errore di tipo `DivError`. Scrivere i tipi delle tre funzioni e, in seguito, quello della funzione `pipeline`, che prende in input le tre funzioni e restitisce il risultato della divisione o quale tipo di errore ha fatto fallire la pipeline. Inoltre, scrivere un breve frammento di pseudocodice che chiama la funzione `pipeline` e stampa se la funzione ha avuto successo o no. Motivare brevemente il ragionamento seguito.

FOGLIO 2 ▷ 6. Considerando la pipeline all'esercizio precedente, si indichino i tipi delle funzioni `input`, `parse`, `div` e `pipeline` per un linguaggio che supporta le eccezioni, dove `f: AxB -> C` **`throws`** `D,E` indica che la funzione `f` prende in input parametri di tipo `A` e `B`, ritorna un valore di tipo `C` e può lanciare eccezioni di tipo `D` ed `E`. Scrivere i tipi delle quattro funzioni e un breve frammento che usa il costrutto **`try-catch`** per chiamare la funzione `pipeline` e stampare se la funzione ha avuto successo o il tipo di eccezione, in caso di fallimento. Inoltre, indicare come avviene la propagazione dell'eccezione lanciata dalla funzione `input` in `pipeline` – può convenire scrivere lo pseudocodice del corpo di `pipeline`. Motivare brevemente il ragionamento.

FOGLIO 2 ▷ 7. Argomentare brevemente differenze, similitudini, vantaggi e svantaggi tra le tecniche basate su tombstones, lock-and-keys, reference counting e stop-and-copy.

FOGLIO 2 ▷ 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 polimorfismo ad-hoc e 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 {
  f() { return this.x; }

  m(n) {
    if (n == 0) return;
    print(this.f());
    this.y = this.x + this.y;
    this.g(n - 1);
  }
}
```

```
object B {
  g(n) {
    if (n == 0) return;
    print(this.f(0));
    this.x = this.x + this.y;
    this.m(n - 1);
  }
  m(n){ this.g(n); },
  f(y) {
    return this.y;
  }
}

object D { x = 1, y = 0 }

A prototypeOf D;
B prototypeOf A;

D.m(5);
```