Corso di Linguaggi di Programmazione
Prova scritta del 2 Luglio 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. Per quali valori delle variabili $X$ e $Y$ la seguente espressione

$$\mathcal{I}_X^{L_0}(\mathcal{C}_{Y,L_1}^{L_1},\mathcal{I}_{L_1}^{L_0})$$

ha senso? Se si, calcola qualcosa di utile?

FOGLIO 1 ▷ 2. Si definisca una grammatica che generi il linguaggio $L=\{a^n b^m c^m d^n \mid n\geq 0, m\geq 1\}$. Tale linguaggio è regolare? Motivare la risposta.

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

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

($i$) 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 $\geq 2$.

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

FOGLIO 2 ▷ 5. Si discutano brevemente i vantaggi e gli svantaggi delle regole di scoping statico e dinamico.

FOGLIO 2 ▷ 6. Si consideri il seguente frammento in uno pseudolinguaggio con scope dinamico e parametri di ordine superiore:

```
{void foo (int f(), int n){
    int m = 10;
    int fie(){
      write(n,m);
    }
    if (n==0) f();
    else {m = 30;
          foo(fie,0);
      }
 }
 int g(){
    write(10);
 }
 foo(g,1);
}
```

Si dica cosa stampa il frammento con deep binding e quali sono le modalità implementative (della regola di binding) che permettono di ottenere tale risultato.

FOGLIO 3 ▷ 7. In uno pseudolinguaggio, `new` crea un nuovo oggetto nello heap. Le strutture A, B e C hanno tutte un campo `next`, a parte B, che ha anche un campo `alt`. Gli oggetti di tipo A, B e C occupano rispettivamente 2, 4 e 6 byte — dato riportato, come `int`, dal rispettivo campo `size`. Il linguaggio usa stop-and-copy, attivando la collection, in funzione delle operazioni di modifica delle strutture in memoria, quando l'occupazione della memoria allocabile supera l'80%. Lo heap ha 30 byte complessivi (inizialmente vuoti). Spiegare brevemente quante volte viene chiamato il garbage collector nell'esecuzione del seguente codice.

```
l(a) {
 if(a.next!=null){return l(a.next);}
 else {return a;} }
h( a ) {
 c = new C();
 c.next = new A();
 l(a).next = c;
 a = g(a);
 if(c( a ) < 3) {h( a );}
}
a = new A();
a.next = new B();
a.next.next = new A();
a.next.alt = new C();
a = g(a);
h(a);
```

```
c(a) {
 if(a != null && a.size==2) {
  return 1 + c(a.next);
 } else {
  return 0;
 }
}

g(a) {
 if(a.next != null){
  if(a.next.size==2){a.next=g(a.next);}
  else{a.next = g(a.next.next);}
 }
 return a;
}
```

FOGLIO 3 ▷ 8. Dare la definizione, più precisa possibile, di polimorfismo universale. Nelle implementazioni di tale meccanismo, che differenze pratiche ci sono tra monomorfizzazione e cancellazione di tipi (type erasure)?