Corso di Linguaggi di Programmazione
Prova scritta del 4 Giugno 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. Si definisca una grammatica che generi il linguaggio $L = \{a^{k+1}b^n c^k \mid k \geq 0, n \geq 1\}$. È regolare tale linguaggio?

FOGLIO **1** ▷ 2. Si consideri la seguente grammatica $G$ con simbolo iniziale $S$:

$$\begin{array}{lcllcl} S & \to & A \mid CB & A & \to & C \mid D \\ B & \to & \texttt{b} \mid \texttt{b}B & C & \to & \texttt{c}C \mid \texttt{c} \\ D & \to & \texttt{d}D \mid \texttt{d} & & & \end{array}$$

($i$) Si calcolino i First e i Follow per tutti i nonterminali. ($ii$) La grammatica $G$ contiene simboli inutili? ($iii$) Si rimuovano le produzioni unitarie da $G$ per ottenere una grammatica $G'$ senza produzioni unitarie equivalente a $G$. ($iv$) La grammatica $G'$ contiene simboli inutili?

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

$$S \to \texttt{abc}S \mid \texttt{abd}S \mid \epsilon$$

($i$) Dimostrare che $G$ è di classe LL(3). ($ii$) Quale linguaggio genera $G$? ($iii$) Manipolare $G$ per ottenerne una equivalente di classe LL(1).

FOGLIO **1** ▷ 4. Costruire un parser bottom-up (shift-reduce) per la grammatica $G$ del punto 3.

FOGLIO **2** ▷ 5. Dire, motivando la risposta, se un linguaggio Turing completo può avere la gestione della memoria completamente statica.

FOGLIO **2** ▷ 6. Si consideri il seguente frammento di codice:

```
int x = 2;
int A[5];
int i;
for (i=0, i<5, i++) A[i]=i;

int fie(int name w,z){
     x =  z + (w++) + (w++);
     write(x)
}

fie(x,A[x]);
```

Si dica qual'è il valore stampato e quali assunzioni sono necessarie per garantire tale risultato.

FOGLIO 3 ▷ 7. Si consideri il programma sotto, scritto in uno pseudolinguaggio dove `T*` è il tipo puntatore ad un valore di tipo `T` e `&x` e `*y` rispettivamente referenzia il valore di `x` e dereferenzia il puntatore `y`. Per ciascuna riga evidenziata con un commento `// [N]`, indicare se la riga presenta un errore nella gestione dei puntatori, specificandone la tipologia, con una breve spiegazione.

```
int* f( int* y ) {
    int x = *y + 42;
    return &x;          // [1]
}

int main() {
    int* q;             // [2]
    q = 42;             // [3]
    q = f(q);
    int x = *q;         // [4]
    free(q);            // [5]
    int* s = NULL;      // [6]
    *s = 10;            // [7]
}
```

FOGLIO 3 ▷ 8. In un pseudolinguaggio, **new** crea un nuovo oggetto nello heap. La lista di nodi `ListLong` occupa 1 byte per ogni nodo della lista. Un `Long` occupa 4 byte. `ListLong` offre `add`, che aggiunge un elemento in coda alla lista, `get`, che ritorna il nodo della lista alla posizione intera passata come parametro, e `remove`, che rimuove dalla lista il nodo alla posizione intera passata come parametro, in modo che l'ultimo nodo della lista è quello prima di quello rimosso. Il pseudolinguaggio usa passaggio per riferimento e garbage collection con contatore dei riferimenti. Indicare se il programma riesce ad eseguire con un heap disponibile di 100 byte, spiegando brevemente il ragionamento seguito.

```
ListLong acc = new ListLong();
for (int d = 4; d <= 8; d = d + 2) {
  int h = 1 + d/2;
  for (int j = 0; j <= d; j++) {
    acc.add(new Long(j));
    if( j == d ){
      acc.add( acc.get( h ) );
      acc.remove( h );
    }
  }
}
```