Corso di Linguaggi di Programmazione — Paradigmi di Programmazione
Prova scritta del 5 luglio 2022.

Tempo a disposizione: ore 2.
Svolgere gli esercizi 1–4, 5–6 e 7–8 su tre fogli differenti.

1. Se L è regolare ed R è libero deterministico, il linguaggio $L \cap \overline{R} = \{w \in A^* \mid w \in L \wedge w \notin R\}$ è regolare o libero, oppure non libero? Giustificare la risposta.

2. Si consideri l'espressione regolare $e = a^*(a|\epsilon)(b|b^*)$. Si verifichi se e sia equivalente all'espressione regolare $d = a^*b^*$, ovvero se $L[e] = L[d]$. Costruire un NFA per d, secondo la costruzione vista a lezione.

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

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

   (i) Verificare che G non è di classe LL(1). (ii) Manipolare la grammatica G, rimuovendo prima le produzioni epsilon e poi le produzioni unitarie. (iii) Quale linguaggio genera G? (iv) Il linguaggio $L(G)$ è di classe LL(1)? Giustificare la risposta.

4. Verificare che il linguaggio $L = \{a^{3n}b^{2n} \mid n \geq 0\}$ è di classe SLR(1).

5. Nel linguaggio di programmazione Pippo sono presenti due comando della forma `lswap N1, N2` e `assign N1, N2` dove `N1` e `N2` sono due nomi che devono essere dotati di l-valore. L'esecuzione di `lswap N1, N2` scambia gli l-valori di `N1` e `N2` mentre `assign N1, N2` assegna l'l-valoriedi `N1` a `N2` Cosa stampa il seguente frammento di codice?

```
int x = 1;
int y = 5;
int z = 100;
lswap x,y;
assign x, z
x++;
y = z+10;
lswap x,y;
write(x,y);
```

6. Si consideri un linguaggio con scope statico, implementato mediante display, nel quale gli identificatori sono noti staticamente. Si assuma inoltre che ogni nome usato sia dichiarato in un solo blocco. Si prendano in considerazione le operazioni di (NL)="accesso ad una variabile non locale x" e (L)="accesso ad una variabile locale y" (nel contesto di un blocco). Per ognuna delle due operazioni di dica a quale dei seguenti parametri è proporzionale il tempo necessario all'esecuzione dell'operazione, motivando brevemente la risposta.

   (i) Il numero di nomi presenti nel programma; (ii) il numero di variabili presenti nel programma; (iii) il numero di blocchi presenti nel programma; (iv) il numero di record di attivazione presenti sulla pila e compresi tra quello contenente la dichiarazione della variabile e quello in cui vi si accede; (v) il numero di blocchi che contengono testualmente il blocco in cui si accede alla variabile e che sono contenuti in quello nel quale la variabile è dichiarata; (vi) Il tempo è costante, e quindi indipendente dai parametri elencati in precedenza; (vii) altro.

7. Usando uno pseudolinguaggio che usi puntatori, si fornisca un esempio di frammento di codice che genera un "dangling reference". Si spieghi, tramite l'esempio, il funzionamento della tecnica delle "tombstones", le garanzie che da e i possibili svantaggi.

8. Si consideri il seguente frammento di codice Java

```
class A {
 double x = 3.0;
 double f(){ return x; }
 double g(){ return f(); }
}
class B extends A {
 double x = 2.0;
 double f( double x ){ return x / 10; }
 double g( double x ){
  return f( f( g() + f() ) ) + f() + f( g() );
 }
}
class C extends B {
 double g(){ return f() - x; }
}

B c = new C();
write( c.g( c.x ) );
```

   Indicare l'output della funzione `write`, che accetta un `double` e lo stampa a schermo nel formato `parte_intera.parte_decimale`. Spiegare brevemente il ragionamento seguito e dare una possibile rappresentazione delle vtable corrispondenti alle classi.