CORSO DI LINGUAGGI DI PROGRAMMAZIONE — PARADIGMI DI PROGRAMMAZIONE
PROVA SCRITTA DEL 6 SETTEMBRE 2023.

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. Descrivere le regole di semantica operazionale strutturata per l'espressione booleana b_0 `xor` b_1, secondo la disciplina di valutazione interna-destra (ID). Ricordo che `xor` è l'operatore di or-esclusivo: l'espressione b_0 `xor` b_1 vale `tt` se e soltanto se il valore di verità di b_0 e di b_1 sono opposti. È possibile definire regole secondo la disciplina esterna-sinistra (ES)?

FOGLIO 1 ▷ 2. Sia data la grammatica G con simbolo iniziale S

$$\begin{array}{lcl} S & \rightarrow & AS|\mathtt{aa} \\ A & \rightarrow & \mathtt{a} \end{array}$$

(*i*) Quale linguaggio genera G? (*ii*) È la grammatica G regolare? (*iii*) Determinare una grammatica G', equivalente a G, in forma normale di Greibach.

FOGLIO 1 ▷ 3. Mostrare che $L_1 = \{a^n b^n a^m \mid n,m \geq 1\}$ è libero deterministico, costruendo un opportuno DPDA. Sapendo che anche $L_2 = \{a^n b^m a^n \mid n,m \geq 1\}$ è libero deterministico, è vero che $L_1 \cap L_2$ è un linguaggio libero deterministico?

FOGLIO 1 ▷ 4. Si consideri la seguente grammatica G con simbolo iniziale S:

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

(*i*) Si verifichi che G è ambigua. (*ii*) Si modifichi al minimo G in modo da ottenere una grammatica G' equivalente non ambigua. (*iii*) Si discuta se la grammatica risultante G' sia di classe LL(1).

FOGLIO 2 ▷ 5. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e deep binding:

```
int x = 3;
procedure ass_x(n:int)
    {x = n;
    }
procedure stampa_x
    {write_integer(x);
    }
procedure pippo(function S, P ; int n )
   { int x= 10;
    if n=1  then ass_x(n)
            else S(n);
    P;
    stampa_x
    }

pippo(ass_x, stampa_x, 1);
pippo(ass_x, stampa_x, 2);
```

FOGLIO 2 ▷ 6. Si consideri la seguente definizione di funzione

```
int f(int n, int m){
   if (n==0) return 1;
   else {
          m = m+1;
          return f(n-1, m+1)
          }
```

Qual è il numero minimo di RdA che una macchina astratta deve usare nel corso della valutazione di `f(5,0)`? Perchè?

FOGLIO 3 ▷ 7. Si indichi quali dei sistemi di equazioni di tipo riportati sotto è risolvibile, indicato uno dei possibili assegnamenti leciti delle variabili di tipo. Le equazioni usano la notazione `Int`, `Str`, per i tipi base, `X`, `Y` per le variabili di tipo e `X -> Int` e `X -> Int -> Y` per indicare il tipo funzione. Per la risoluzione delle equazioni può essere utile ricordare la logica dell'algoritmo di unificazione. Ad esempio, il sistema di equazioni `X -> X -> X = Int-> Y, X -> X = Y` ha soluzione `X = Int, Y = Int -> Int`.

S1. `X = Nat, Y = X -> X`

S2. `Nat = Z -> X`

S3. `X -> X -> X = Int -> Y, X -> X = Y`

S4. `X -> Y = Y -> Z, Z = Int -> Str`

FOGLIO 3 ▷ 8. Si consideri un linguaggio con passaggio per valore nel quale le eccezioni: devono essere dichiarate con la sintassi `exception E1, ..., En` con `E1 ... En` nomi di eccezione; vengono sollevate con l'istruzione `throw E`; vengono gestite coi blocchi `try { ... } catch E1 { ... } ... catch En { ... }`. Il linguaggio ha scoping statico per tutti i nomi, eccezioni comprese. Cosa stampa (tramite l'operazione `print`) il seguente frammento? Spiegare brevemente il ragionamento dietro la risposta.

```
exception E1, E2, E3;

int a = 6; int b = 0;

d() {
 if ( a < 8 ) { a++; throw E2; }
 else { throw E3; }
}

c() { b++; throw E1; }

b( s ) {
 try {
    if( s.length() % 2 == 0 ){  d();  }
    else { c(); }
 } catch E3 { print( s + a ); }
}

a( s ){
 try{ b( s ); }
 catch E1 { a( s + b ); }
 catch E2 { a( s + a ); }
}

a( "2." );
```