CORSO DI LINGUAGGI DI PROGRAMMAZIONE
PROVA SCRITTA DEL 4 GIUGNO 2024.

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. Le seguenti due definizioni di grammatica libera da contesto sono equivalenti?

(a) $G = (NT, T, S, R)$ è libera da contesto se ogni produzione in R ha la forma $A \to \alpha$, con $A \in NT$ e $\alpha \in (T \cup NT)^*$.

(b) $G = (NT, T, S, R)$ è libera da contesto se ogni produzione in R ha la forma $A \to \alpha$, con $A \in NT$ e $\alpha \in (T \cup NT)^+$, ovvero $\alpha \neq \epsilon$. Tuttavia, è ammessa la produzione epsilon per il simbolo iniziale S ($S \to \epsilon$), ma in tal caso S non può comparire nella parte destra di nessuna produzione in R.

FOGLIO 1 ▷ 2. Si costruisca l'automa più semplice che riconosca il linguaggio $L = \{a^n \mid n = 4k + 2, k \geq 0\}$.

FOGLIO 1 ▷ 3. Costruire un DPDA che riconosca il linguaggio $L_2 = \{a^{n+1} c b^n \mid n \geq 0\}$.

FOGLIO 1 ▷ 4. Costruire un parser $LR(0)$ per il linguaggio $L = \{a^n b^n \mid n \geq 1\}$ e si discuta il suo comportamento sugli input *aabb* ed ϵ.

FOGLIO 2 ▷ 5. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e passaggio di parametri per nome e per riferimento.

```
{int x = 1;
 int y = 5;
 int z = 10;
void pippo(name int y, reference int z){
     z = x + y + z;
     }
   { int x = 20;
     int y = 30;
     int z = 50;
     pippo(x++, x);
     pippo(x++, x);
     write(x);
     }
 write(x);
 }
```

(la primitiva `write(x)` permette di stampare un valore intero; un comando della forma `foo(w++);` passa a foo il valore corrente di `w` e poi incrementa `w` di uno). Fare opportune ipotesi sulla valutazione delle espressioni, descrivendo brevemente il ragionamento fatto.

FOGLIO 2 ▷ 6. Si scriva, in un qualsiasi pseudo-linguaggio, un frammento di codice tale che il numero dei record di attivazione (RdA) presenti a run-time sulla pila fra il RdA di una chiamata di procedura e quello del blocco che la (sintatticamente) la contiene, vari a seconda del valore di una variabile letta dall'esterno.

FOGLIO 3 ▷

7. Considerando il frammento di codice sotto, scritto in uno pseudo-linguaggio con sistema di tipi dinamico (duck typing), indicare quali istruzioni (I1–I5) vengono segnalate come errate, spiegando brevemente il motivo. Infine, assumendo di avere un sistema di tipi statico, definire il tipo della variabile `x`. Il linguaggio supporta record e unioni con le risp. sintassi `{a: int, b: ..., ...}` e `A | ... | C`. La definizione del tipo di `x` può usare definizioni di tipi ausiliari, ad es., il tipo `A` che all'interno usa `B` e/o `C`. Nel linguaggio, `{ a: 1, b: 2 }` è un valore di tipo record e `{}` è un record vuoto.

```
j( y ){
 y.d = { a: 5, b: { c: { a:6, e: 7 }, d: { e: 8 } } }
 return y
}
h( y ){
 y.c = { e: 4 }
 return y
}
x = { a: 1, b: j( h( {} ) ) }
x.b.c.b      // I1
x.c          // I2
x.a          // I3
x.b.b        // I4
x.b.d.a      // I5
```

FOGLIO 3 ▷

8. È dato un linguaggio con passaggio per riferimento ed eccezioni dichiarate con **exception** `E` (`E` nome dell'eccezione), sollevate con l'istruzione **throw** `E` e gestite coi blocchi **try** `{ ... }` **catch** `E { ... }`. Il tipo `List` offre le operazioni: `of` per creare una lista di elementi del valore e ordine dei parametri dati, `add` per aggiungere il parametro in testa alla lista, `remove` per rimuovere e ritornare l'elemento in testa alla lista e `isEmpty`, che ritorna **true** solo se la lista è vuota. Dando a `print` un `List`, questa stampa gli elementi della lista in ordine, separati da virgole. Cosa stampa il frammento sotto? Spiegare brevemente la risposta.

```
Exception Z;
d( i, n ) {
 if ( i % n == 0 ) { throw Z; }
}
Exception X;
c( s, i, n ){
 try { b( s, n ); } catch ( X ) {}
 s.add( i );
}
b( s, n ) {
 if( s.isEmpty() ){ throw X; }
 i = s.remove();
 try { d( i, n ); c( s, i, n ); }
 catch ( Z ) { b( s, n ); }
}
```

```
Exception Y;
a( s ) {
 if( s.isEmpty() ){ throw Y; }
 n = s.remove();
 try { b( s, n ); } catch ( X ) {}
 try { a( s ); } catch ( Y ) {}
 s.add( n );
}

l = List.of( 2,8,6,3,5,12,7,11,14,10 );
a( l );
print( l );
```