19 aprile 2007

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	
Telefono	
Email	

1. Specificare la grammatica BNF del linguaggio **L** in cui ogni frase è una grammatica (non vuota) EBNF in forma non fattorizzata (un'unica alternativa per ogni nonterminale), utilizzando per **L** (tra gli altri) i seguenti terminali: **nonterminal** (per identificare un nonterminale), **terminal** (per identificare un terminale), **epsilon** (per identificare il simbolo **ε**). Si assumono per le frasi di **L** unicamente i seguenti operatori estesi:

   - { … } = ripetizione zero o più volte;
   - […] = opzionalità.

   (Si noti che non si considera l'operatore alternativa).

2. È data la seguente tabella di operatori, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
=	binario	-	da sinistra a destra	no
not	unario	destra	-	-
and	binario	sinistra	da sinistra a destra	si
or	binario	sinistra	da destra a sinistra	no
?:	ternario	-	da sinistra a destra	si

   e la seguente espressione:

```
not a = b and c or d = e ? x and y : z
```

   a) Rappresentare l'albero della espressione.
   b) Specificare la semantica operazionale della espressione.

   NB: Il linguaggio di specifica operazionale è così caratterizzato:

   - Contiene gli operatori di negazione (**!**), disgiunzione (**||**), uguaglianza (**==**) ed assegnamento (**:=**);
   - Non contiene l'operatore di congiunzione, ne l'operatore **?:**, ne le costanti logiche **true**, **false**;
   - Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
   - Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati possono essere solo variabili;
   - Contiene l'istruzione **return** il cui argomento è la variabile che contiene il valore della espressione;
   - L'esecuzione della **return** termina immediatamente l'esecuzione del programma di specifica operazionale.

3. Dopo aver definito in *Scheme* la funzione booleana `appartiene`, che stabilisce se `x` appartiene alla lista `set`, come nei seguenti esempi,

x	set	(appartiene x set)
1	(1 2 3)	#t
4	(a b c)	#f
(a)	(a b c)	#f
(b)	(a (b) c)	#t
()	()	#f
()	(1 2 ())	#t

   definire la funzione `intersezione`, avente in ingresso due liste (senza duplicati), `set1` e `set2`, che computa l'intersezione insiemistica `set1` ∩ `set2`.

4. Definire nel linguaggio *Haskell* la funzione `all`, che riceve una lista di elementi ed una funzione booleana `f` avente come dominio lo stesso tipo degli elementi di `lista`. La funzione `all` restituisce `True` se tutti gli elementi di `lista` rendono vera la `f`, altrimenti restituisce `False`. (Nel caso in cui `lista` è vuota, `all` retituisce `True`).

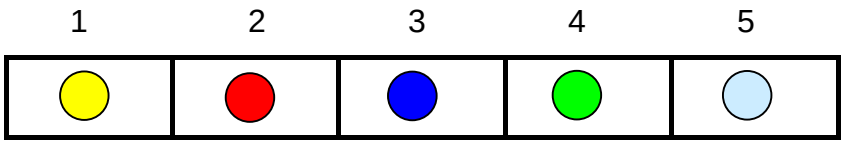
5. È data una scatola costituita da cinque scomparti, numerati (da sinistra a destra) 1, 2, 3, 4 e 5, e cinque palline, di colore giallo, rosso, blu, verde e azzurro. Specificare in *Prolog* il predicato

```
scatola(P1, P2, P3, P4, P5)
```

   in cui `Pi` è il colore della pallina nell'`i`-esimo scomparto, che risulta vero se le palline sono allocate nei diversi scomparti secondo i seguenti vincoli:

   - Le palline rossa e verde non sono allocate nel terzo scomparto;
   - La pallina blu non è allocata ne nel primo ne nell'ultimo scomparto;
   - La pallina blu è immediatamente preceduta (a sinistra) da quella rossa e immediatamente seguita (a destra) da quella verde;
   - La pallina gialla è allocata a sinistra (non necessariamente nella posizione adiacente) della pallina blu;
   - La pallina azzurra segue immediatamente la pallina verde.

   Ecco una possibile soluzione (`P1` = giallo, `P2` = rosso, `P3` = blu, `P4` = verde, `P5` = azzurro):



6. Nell'ambito del paradigma orientato agli oggetti, definire e giustificare (sulla base di un semplice esempio) la regola di covarianza del parametro di uscita nei metodi.