Linguaggi di Programmazione (III Anno), Luglio 1998 - OOP

Si puo' usare una sintassi sia C++ che Java-like (ma si consiglia l'uso di Java). Per le parti di codice che sono indipendenti dagli aspetti object-oriented di tali linguaggi si consiglia di usare pseudo-istruzioni o una descrizione a parole.

Esercizio OOP.1 (8 punti) Si dica qual'e' l'output del seguente programma Java-like (si consiglia di descrivere brevemente i criteri generali per scegliere il metodo).

```
class Parent {
int m (Parent p) { return 0; }
int n (Heir h) { return (h.m(h) + 2);}
}
class Heir extends Parent {
int m (Heir h) { return 2; }
int n (Heir h) { return (super.n(h) + 1);}
}

Parent p1 = new Parent ();
Parent p2 = new Heir ();
Heir h = new Heir();

print(p2.m(p1)); print(p2.m(p2)); print(h.m(p1)); print(h.m(h));

print(p1.n(h)); print(p2.n(h)); print(h.n(h));
```

Esercizio OOP.2 (8 punti) Si implementino le sottoclassi *point* e *shifted_set* della classe astratta *set_of_points* (insieme di punti della retta reale). *shifted_set* consiste di una lista l di *set_of_points* tutti traslati di una quantita' x.

```
abstract class set_of_points {
abstract public move(real d);  TRASLA I PUNTI DI d
abstract public scale(real s); SCALA DISTANZE DEI PUNTI DALL'ORIGINE DI s
abstract public bool in (point p);    CHECK SE p E' NELL'INSIEME
}
class list_of_sets {
public set_of_points first;
public list_of_sets others;
}
class point extends set_of_points {
...                          PARTE PRIVATE
public point(real x)         DA IMPLEMENTARE (COSTRUTTORE)
public real x()              DA IMPLEMENTARE
METODI DI set_of_points      DA IMPLEMENTARE
}
class shifted_set extends set_of_points {
...                          PARTE PRIVATE
public shifted_set(real x, list_of_sets l)
                             DA IMPLEMENTARE (COSTRUTTORE)
METODI DI set_of_points      DA IMPLEMENTARE
}
```

Linguaggi di Programmazione (III Anno), Luglio 1998 - FP

Esercizio FP.1 (4 punti) Dire se le seguenti espressioni sono tipabili in ML (usando l'algoritmo W di Hindley-Milner), e in caso affermativo dare il tipo piu' generale.

1. $\lambda f, x.cons(x, f\ x)$
2. $\lambda f, x.cons(f\ x, x)$

dove $cons : \forall X.X \to X\ list \to X\ list$.

Esercizio FP.2 (6 punti) Si consideri il tipo di dato `set_of_points`, i cui elementi rappresentano insiemi di punti sulla retta:

```
datatype set_of_points =
    point of { x:real }
  | shifted_set of { x:real, L:set_of_points list };
```

In particolare `shifted_set {x=a, L=[f1,...,fn]}` e' un lista di insiemi di punti `f1,...,fn` tutti shiftati di x (SI DARA' QUALCHE ESEMPIO ALLA LAVAGNA). Si definiscano le operazioni $shift, scale : real \to set_of_points \to set_of_points$, che su un singolo punto si comportano nella seguente maniera:

```
fun shift d (point {x=a}) = point {x=a+d}
  | ...
fun scale s (point {x=a}) = point {x=a*s}
  | ...
```

(si suppone che $s > 0$).

Esercizio FP.3 (6 punti) Dato il tipo "alberi binari", definito come segue

```
datatype 'a btree = empty | node of 'a * 'a btree * 'a btree;
```

si consideri la seguente funzione higher-order:

```
fun btree_it f a empty = a
  | btree_it f a (node(b,t1,t2)) = f(b, btree_it f a t1, btree_it f a t2);
```

- Si indichi il tipo piu' generale di `btree_it`.
- Si definiscano istanziazioni di `btree_it` che calcolano:
  il numero di nodi di un albero,
  la lista dei nodi secondo la visita inorder