PROGRAMMAZIONE II (A,B) - a.a. 2019-20
Primo Appello – 7 Febbraio 2020

Domande di base

1. Si consideri il seguente frammento di programma

```
class A{ }
class B extends A{}
class C extends A{}
class D extends B{}

\\ main
A x = new B();
B y = new D();
x=y;
A w = new C();
```

   Definire il tipo statico e quello dinamico per ogni variable presente nel `main`.

2. Descrivere la struttura e le informazioni presenti nel record di attivazione dell implementazione di un linguaggio con le caratteristiche di OCaML.

3. Spiegare il motivo per cui la regola di scoping dinamico comporta la necessità di avere un meccanismo di type cheching dinamico.

Esercizio 1

Si consideri un tipo di dato astratto *OrderedRelation* introdotto per rappresentare una *relazione binaria* tra un dominio di oggetti generici e un codominio di oggetti generici che hanno associata una nozione di ordinamento. Tutti gli elementi del codominio associati ad un dato elemento del dominio sono disposti in ordine crescente. L'interfaccia generica `OrderedRelation<E,T extends Comparable<T>>` include, tra gli altri, i seguenti metodi

- `public void add(E x, T y)` *il cui effetto è quello di aggiungere la coppia (x,y) alla relazione*
- `public void remove(E x, T y)` *il cui effetto è quello di rimuovere la coppia (x,y) dalla relazione*
- `public void removeAll(E x)` *il cui effetto è quello di rimuovere tutte coppie di valori che sono associate ad x nella relazione*
- `public int number(E x)` *restituisce il numero di valori associati ad x presenti nella relazione*
- `public int size()` *restituisce il numero di coppie presenti nella relazione*
- `public T minElem(E x)` *il cui effetto è quello di restituire l' elemento associato ad x nella collezione che è il minimo tra tutti gli elementi associati ad x*

1. Si definisca la specifica dell'interfaccia `OrderedRelation<E,T extends Comparable<T>>`, indicando l'elemento tipico e per ogni metodo le clausole REQUIRES, MODIFY ed EFFECTS, il valore restituito e le eventuali eccezioni lanciate in dipendenza dei parametri attuali.

2. Si assuma di implementare l'interfaccia `OrderedRelation<E,T extends Comparable<T>>` con una classe concreta che ha le seguenti variabili d'istanza:

```
private  HashMap<E, ArrayList<T>> rel;
```

(a) Si definiscano la funzione di astrazione e l'invariante di rappresentazione.

(b) Si implementino il costruttore ed i metodi `add` e `minElem` e si dimostri che l'implementazione proposta preserva l'invariante di rappresentazione.

3. Si consideri una classe `OrderedFun<E,T extends Comparable<T>>` in cui le coppie della relazione formano una funzione. La classe `OrderedFun<E,T extends Comparable<T>>` può essere sottotipo di `OrderedRelation<E,T extends Comparable<T>>` secondo il principio di sostituzione di Liskov? Motivare la propria risposta.

Esercizio 2

Si consideri il seguente programma OCAML

```
let z =3;;
let f1= fun x -> fun y ->  x+y-z;;
let g1= f1 10;;
let rec apply f n x =
  if n<=0 then x
  else apply  f (n-1) (f x);;
let rec map  g n = function
  | [] -> []
  | h1::ls -> (apply  g n h1) :: map g (n-1) ls;;
let z = 1;;
map g1 (z+1)  [1;20;3];;
```

Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.

Esercizio 3

1. Estendere il linguaggio didattico funzionale con il tipo di dato astratto `IntTree` per poter operare con alberi binari con nodi contenenti valori di tipo intero. Per esempio in sintassi concreta

```
let t0= Empty;;
let t1=Node(Node(Empty, 1, Empty),10, Empty);;
```

2. Introdurre una nuova tipologia di astrazione funzionale `filter-by` che consente di applicare una funzione (non ricorsiva) a tutti gli elementi di un albero binario che appartengono ad un certo intervallo. Per esempio in sintassi concreta l'astrazione `filter-by(fun x= x+1,[0,5])` applicata a `t0` produce come risultato l'albero binario `t0` mentre l'astrazione `filter-by(fun x= x+1,[0,5])` applicata a `t1` produce come risultato l'albero binario `Node(Node(Empty, 2, Empty),10, Empty)`.

   Definire le regole di valutazione dell'astrazione `filter-by` e estendere consistentemente la struttura dell'interprete del linguaggio didattico funzionale.