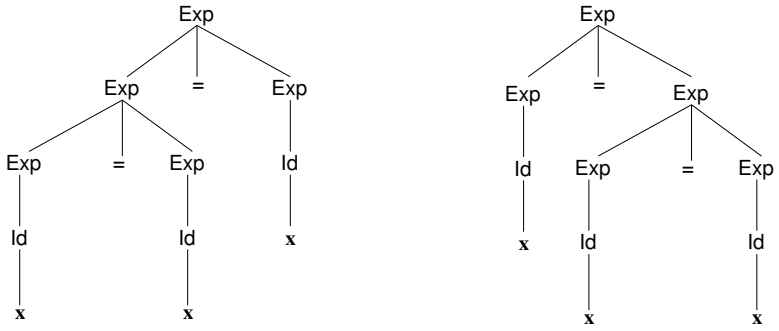
Linguaggi e Programmazione Orientata agli Oggetti

Soluzioni della prova scritta

a.a. 2011/2012

13 luglio 2012

1. (a) Esistono i seguenti due alberi di derivazione per la stringa `x=x=x`.



(b)

```
Exp  ::= Exp2 = Exp | Exp2
Exp2 ::= Id | Bit | ( Exp )
Id ::= x | y | z
Bit ::= 0 | 1
```

2. (a)
```
let rec tree_member x = function
    Node(y,l) -> x=y || forest_member x l
and forest_member x = function
    [] -> false
  | t::l -> tree_member x t || forest_member x l;;
```

(b)
```
let tree_member x = tree_exists (fun y -> y=x);;
```

(c)
```
let rec count_tree_nodes = function
    Node(_,l) -> 1 + count_forest_nodes l
and
    count_forest_nodes = function
        [] -> 0
      | t::l -> count_tree_nodes t + count_forest_nodes l;;
```

Oppure, con parametro di accumulazione:

```
let count_tree_nodes = aux_count_tree_nodes 0;;

let rec aux_count_tree_nodes n = function
    Node(_,l) -> aux_count_forest_nodes (n+1) l
and
    aux_count_forest_nodes n = function
        [] -> n
      | t::l ->  aux_count_forest_nodes (aux_count_tree_nodes n t) l;;
```

3. (a)
```
public TreeClass(E elem) {
    root = new Node<>(elem);
}
```

(b)
```
private boolean contains(E elem) {
    if (this.elem == elem)
        return true;
    for (Node<E> node : children) {
        if (node.contains(elem))
            return true;
    }
    return false;
}
```

(c)
```
@Override
public boolean contains(E elem) {
    return root.contains(elem);
}
```

(d)
```
@Override
public E get(Stack<Integer> index) {
    Node<E> node = root;
    while (!index.isEmpty())
        node = node.children.get(index.pop());
    return node.elem;
}
```

(e)
```
@Override
public E set(Stack<Integer> index, E elem) {
    Node<E> node = root;
    while (!index.isEmpty())
        node = node.children.get(index.pop());
    E oldElem = node.elem;
    node.elem = elem;
    return oldElem;
}
```

4. (a) Errore di compilazione: il metodo `m` non è visibile in `C2`.

(b)
```
C3.m
C4.m
```

(c)
```
C2.q
C1.q
C2.m
```

(d)
```
C3.m
```

(e)
```
C2.q
C1.q
C2.m
```

(f)
```
C4.q
C2.q
C1.q
C3.m
C4.m
```