Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
9 settembre 2019

Esercizio 1.A – rispondere, in maniera concisa, alle seguenti domande (12 punti)

1. Qual'è l'uso del costrutto "foreach"?

2. Nel passaggio dei parametri per nome, cosa si intende per chiusura?

3. Nei linguaggi funzionali, come differisce la valutazione eager da quella lazy?

4. Dal punto di vista dei sistemi di tipi, in cosa differiscono Haskell e Scheme?

5. Cos'è un lessema?

6. Quali sono i vantaggi della compilazione rispetto all'interpretazione?

7. Cosa sono i tipi enumerazione?

8. Presentare un esempio di polimorfismo ad hoc.

9. Quali sono le problematiche dell'ereditarietà multipla?

10. Nella programmazione concorrente, cosa sono i canali?

11. Nei comandi in assegnazione, cosa si intende per R-value?

12. Cosa si intende per calcolo vettoriale?

13. Che tipo di dati vengono memorizzati nell'heap?

14. Cosa distingue i linguaggi class based dai linguaggi prototype based?

15. Cosa produce l'analisi sintattica di un programma?

16. Elencare alcune motivazioni per la programmazione concorrente.

17. Cosa prevede il principio dell'incapsulamento?

18. Nello scoping statico, come si determina il binding valido?

1. Cognome e Nome:_____ Matr.:_____

– 9 settembre 2019– A

Esercizio 2.A – Grammatiche (4 punti)

1. Si consideri la grammatica (con P simbolo iniziale):

$$P \rightarrow \epsilon \mid aP \mid bD$$

$$D \rightarrow b \mid aD \mid bP$$

- si scrivano tutte le parole di lunghezza minore di 6 generati dalla grammatica,
- si stabilisca se la grammatica è ambigua,
- nel caso lo sia, si definisca una grammatica non ambigua equivalente,
- si descriva il linguaggio generato,
- si stabilisca se è un linguaggio regolare,
- nel caso lo sia, lo si descriva come espressione regolare.

Esercizio 3.A – Stack di attivazione (6 punti)

1. Si mostri l'evoluzione dello stack di attivazione e dell'output dei due frammenti di programma seguenti. Si ipotizzi che il linguaggio C-like abbia scoping dinamico, deep binding, assegnamento che calcola r-value prima di l-value, valutazione delle espressioni da sinistra a destra, valutazione degli argomenti da sinistra a destra, e indici vettori inizianti da 0:

```
int i=1, j=2, v[3]={3,4,5};
int foo(ref int i, val int z){
  while ((v[i--] += v[i]) < 20){
    z = v[i]++ + (v[j++]);
    write (v[i],z);};
  return i;
};
write(foo(j, v[j--]));
write(v[i],j);
```

```
int x = 4, y = 8;
int F(name int v, val int y){
  y += v;
  write(v, x, y)
  return(v + x);
};
{
   int x = 2;
   int Q(ref int y, val int v){
     y = F(v++, y);
     write(v, x++, y);
    return (x + v);
   };
   write(Q(y, x);
   write(x, y);
}
write(x, y);
```

1. Cognome e Nome:_____ Matr.:_____

– 9 settembre 2019– A

Esercizio 4.A – Haskell (7 punti)

1. Si scriva una funzione Haskell che, prese in ingresso due liste strettamente ordinate xs e ys, costruisca una terza lista strettamente ordinata contenente tutti gli elementi che appaiono in almeno una delle due le liste xs e ys. Si rispetti perciò il vincolo che se un elemento appare sia in xs che in ys, allora deve apparire un'unica volta nella lista risultato.

   Si scriva inoltre una funzione Haskell che, presa in ingresso una matrice memorizzata per righe, in cui ogni riga è strettamente ordinata, costruisca una lista strettamente ordinata contenente tutti gli elementi che appaiono in almeno una riga della matrice.

   Si commenti il codice e si definisca il tipo di ogni funzione definita.

Esercizio 5.A – Sistema di assegnazione di tipo (4 punti)

1. Nel sistema di tipi per il linguaggio F1, costruire la derivazione di tipo per la seguente espressione:

```
\ f :(Nat -> Bool) -> ( \ n :Nat -> if_Bool (f n) then false else true )
```