Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
17 giugno 2019

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

1. Cosa distingue una comunicazione sincrona da una asincrona?

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

3. Cos'è la programmazione strutturata?

4. In un linguaggio imperativo, cosa sono i valori denotabili?

5. Cosa produce l'analisi lessicale di un programma?

6. Qual'è la differenza tra "coercion" e "casting"?

7. Quali sono le caratteristiche del passaggio dei parametri per nome?

8. Cosa sono le espressioni regolari?

9. Cosa si intende per selezione dinamica dei metodi?

10. A cosa servono le dichiarazione di import-export nei moduli?

11. Nella programmazione concorrente, cosa sono i canali?

12. Cosa sono le funzioni di ordine superiore?

13. A quale scopo vengono introdotte le definizioni di type class in Haskell?

14. Cosa si intende per semantica di un linguaggio di programmazione?

15. Descrivere le regole dello scoping statico.

16. Nello stack di attivazione, cos'è il link statico?

17. Nella rappresentazione delle espressioni, come viene definita la notazione polacca inversa?

18. Perché l'aritmetica sui puntatori può essere problematica?

1. Cognome e Nome:_____ Matr.:_____

– 17 giugno 2019– A

Esercizio 2.A – Grammatiche (4 punti)

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

$$\begin{array}{lll} P & \rightarrow & \epsilon \mid aP \mid bD \\ D & \rightarrow & a \mid aD \mid bP \end{array}$$

   - si scrivano tutte le parole di lunghezza minore di 5 generate dalla grammatica L,
   - si stabilisca se la grammatica è ambigua,
   - si descriva il linguaggio generato,
   - si stabilisca se è un linguaggio regolare,
   - nel caso, 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 statico, assegnamento che calcola prima l'r-value e poi l'l-value, valutazione delle espressioni da sinistra a destra e indici vettori inizianti da 0:

```
int i=2, j=1, v[3]={4,3,5};
int foo(val int i, ref int z){
  while ((v[i--] += v[i]) < 15){
    z = v[++i] + (v[j]++);
    write (v[i]);};
  return i;
};
write(foo(j++, v[j--]));
write(v[i],j);
```

```
int x=1;
int y=0;
void P(valueresult int y, ref int z, int R(name int)){
  z = y++ + R(x + y);
  write(x, y, z);
  z = R(z++);
}
{
  int x =3;
  int Q(name int w){
    return(w + x);
  }
  P(x ,y ,Q);
  write(y,x++);
}
write(y, x);
```

1. Cognome e Nome:_____ Matr.:_____

– 17 giugno 2019– A

Esercizio 4.A – Haskell (7 punti)

1. Si scriva una funzione Haskell che, prese in ingresso due liste ordinate, costruisca una lista ordinata contenente tutti gli elementi che appaiono in entrambe le liste. Si scriva inoltre una funzione Haskell che, presa in ingresso una matrice memorizzata per righe, in cui ogni riga è ordinata, costruisca una lista ordinata contenente tutti gli elementi che appaiono in ogni 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 -> Nat) -> (\x :  Nat -> f ( f x ) )
   ```

   Nota: i primi passi di derivazione, a partire dagli assiomi, se elementari o ripetizioni di derivazioni già fatte, possono essere omessi.