Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
5 luglio 2021

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

1. Elencare alcuni meccanismi di sincronizzazione nella programmazione concorrente.

2. Cos'è un record di attivazione?

3. Cosa sono i moduli?

4. Quali sono i vantaggi della valutazione eager rispetto alla valutazione lazy?

5. Descivere i tipi generics di Java.

6. Cosa sono i tipi enumerazione?

7. Presentare un esempio di polimorfismo ad hoc.

8. In un linguaggio di programmazione, cosa sono i valori denotabili?

9. Cos'è il supporto a run-time di un linguaggio di programmazione?

10. Che tipo di linguaggi vengono riconosciuti dagli analizzatori lessicali?

11. Cosa si intende per scoping statico?

12. Cosa si intende per selezione dinamica dei metodi?

13. Perché un comando "case" può essere più efficiente rispetto ad una serie di comandi if-then-else?

14. Nella rappresentazione delle espressioni, quali sono i vantaggi della notazione polacca inversa?

15. Cosa sono le funzioni di ordine superiore?

16. In un linguaggi di programmazione, cosa s'intende per ambiente?

17. Cosa si intende per mutua esclusione?

18. Quali sono le caratteristiche del passaggio dei parametri per valore/risultato?

1. Cognome e Nome:_____ Matr.:_____

– 5 luglio 2021– A

Esercizio 2.A – Grammatiche (4 punti)

1. Date le seguenti regole di produzione

$$S_1 ::= bS_1 \mid P \qquad S_2 ::= PP$$
$$P ::= a \mid b \mid aPa \mid bPb$$

$L(S_1)$ e $L(S_2)$ indicano i linguaggi generati dalle grammatiche con simboli iniziali S_1 e S_2

- Si diano le stringhe di $L(S_1)$ e $L(S_2)$ di lunghezza ≤ 4;
- si caratterizzino le stringhe di $L(S_1) \cap L(S_2)$;
- si dica se la grammatica G iniziante da S_1 è ambigua (mostrando un testimone dell'ambiguità oppure argomentando opportunamente sulla non ambiguità);
- si stabilisca se il linguaggio $L(S_1)$ è regolare (argomentando opportunamente la risposta), e nel caso, lo si descriva in linguaggio mediante un 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, shallow binding, assegnamento che calcola prima l-value e poi r-value, valutazione delle espressioni da sinistra a destra, valutazione degli argomenti da sinistra a destra, e indici vettori inizianti da 0.

   Inoltre, per il secondo frammento di codice, si mostri anche l'evoluzione del CRT con pila nascosta.

```
int v[3]={5,2,7}, i=2, j=0;
void f(name int j, valres int[ ]w){
  foreach(int x : w)
    write(w[i] += ++v[i]+j+x) ;
}
f(v[j++], v);
write(v[i],j) ;
```

```
int x=3, y=2;
int G(valres int z, int R(ref int)){
  x = R(z);
  write(x, z);
  return(x + z);
}
int F(name int y){
  int x=4;
  int H(ref int w){
    w =+ y;
    return(w + x); }
  return(y+G(x,H) ) ;
}
write (F(++x),x,y);
```

1. Cognome e Nome:_____ Matr.:_____

– 5 luglio 2021– A

Esercizio 4.A – Haskell (7 punti)

1. Scrivere le seguenti funzioni in Haskell:
   - una funzione che data lista determini se questa sia palindroma;
   - un funzione che data una matrice memorizzata per righe determini se questa sia simmetrica rispetto all'asse verticale,
   - un funzione che data una matrice memorizzata per righe determini se questa sia simmetrica rispetto all'asse orizzontale,
   - un funzione che data una lista determini la più lunga sottosequenza iniziale palindroma della lista data.

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

Esercizio 5.A – Equivalenza tra tipi (2 punti)

1. Nelle ipotesi che si usino parole di memoria di 4 byte con memorizzazione a parole allineate e che un intero occupi una parola di memoria ed un carattere un byte, determinare quanti byte di memoria occupano i due seguenti tipi di dato:

```
struct A {
  int i[5];
  char a[6];
  int j;
  char b; }

struct B {
  int i;
  int j;
  char a;
  char b; }
```

Esercizio 6.A – Sistema di assegnazione di tipo (3 punti)

1. (2-7-20) Nel sistema di tipi per il linguaggio imperativo presentato nei lucidi, costruire la derivazione di tipo per il comando:

   `{ Nat i=2; inc(Nat j){i = i+j}; if i<0 then inc(3) else inc(i+3) }`

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