Cognome e Nome:_____ Matr.:_____

Linguaggi di programmazione – A
19 settembre 2018

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

1. Quali sono le caratteristiche di un strong type system?

2. Nella programmazione concorrente, cosa sono i canali?

3. Si mostri un esempio in C in cui l'aritmetica dei puntatori differisce dall'aritmetica standard?

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

5. Cos'è una grammatica libera da contesto?

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

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

8. Quali controlli sulla sintassi non possono essere svolti attraverso le grammatiche libere?

9. Cosa sono i semafori?

10. Come si caratterizza l'ereditarietà singola?

11. Cos'è la programmazione strutturata?

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

13. A cosa servono le eccezioni?

14. Cos'è un interprete di un linguaggio di programmazione?

15. Il programma YACC che tipo di automi genera?

16. Cosa si intende per scoping statico?

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

18. Cosa si intende per equivalenza strutturale tra i tipi?

1. Cognome e Nome:_____ Matr.:_____

– 19 settembre 2018– A

Esercizio 2.A – Grammatiche (4 punti)

1. Si consideri la grammatica:

$$A \rightarrow \epsilon \mid aAbA \mid bAaA$$

   - 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,
   - si abbozzino delle regole YACC-Happy, per la creazione di un programma che riconosca il linguaggio generato dalla grammatica A e dalla stringa costruisca l'albero di derivazione.

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 l-value prima di r-value, valutazione delle espressioni da sinistra a destra e indici vettori inizianti da 0:

```
char s[10] = "abcdefghij" ;
int k = 1 ;
char foo(ref int j, val char c)
{
  write(s[j++], c);
  c = s[++j];
  write(s[j++], c);
  return c;
}
write(foo(k, s[k++]) );
write(k);
```

```
int x = 2 , y = 6 ;
int F (val int v , ref int n) {
  n += v ;
  v += n ;
  write (x, y, v, n)
  return v;
}
int Q (ref int y , int R( val int , ref int ) ) {
  y = R( y++ , y ) ;
  return (y) ;
}
write (Q(x, F)) ;
write (x, y ) ;
```

1. Cognome e Nome:_____ Matr.:_____

– 19 settembre 2018– A

Esercizio 4.A – Haskell (7 punti)

1. Scrivere una funzione Haskell che dato un numero naturale n costruisca una scacchiara di dimensione n ossia una matrice quadrata contenenti i soli valori 0 e 1 alternati.

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

1. Nel sistema di tipi per il linguaggio imperativo presentato nei lucidi, costruire la derivazione di tipo per il comando:

   `{ Nat x = 4; Nat y = x + 4; while (x > 0) {  y = y + x ; x = x - 1 } }`

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