Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
12 settembre 2022

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

1. Cos'è una "race condition"?

2. Descrivere brevemente la notazione polacca diretta?

3. Dal punto di vista del sistema di tipi, quali sono le differenze tra Haskell e Scheme?

4. Come si caratterizza l'ereditarietà singola?

5. Quali sono le caratteristiche del polimorfismo di sottotipo?

6. Nella programmazione concorrente, cosa sono le porte?

7. Quali sono le operazione tipiche sui puntatori?

8. Cosa distingue un comando da un'espressione?

9. Cos'è un compilatore?

10. Cos'è la BNF (Backus Naur Form)?

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

12. Cosa sono i moduli?

13. Quando la definizione ricorsiva di una funzione è tail recursive?

14. Nell´istruzione di assegnazione, cosa si intende per R-value?

15. Come accade quando viene generata un'eccezione?

16. Cosa si intende per scoping statico?

17. Cos'è un thunk?

18. Cosa si intende per memorizzazione per colonne di una matrice?

1. Cognome e Nome:_____ Matr.:_____

– 12 settembre 2022– A

Esercizio 2.A – Grammatiche (4 punti)

1. Data la seguente regola di produzione:

$$T ::= l \mid TnT$$

- Si scrivano le stringhe di $L(T)$ di lunghezza ≤ 8;
- si caratterizzino le stringhe di $L(T)$;
- si dica se la grammatica è ambigua (mostrando un testimone dell'ambiguità oppure argomentando opportunamente sulla non ambiguità);
- nel caso la grammatica sia ambigua, si definisca un'opportuna grammatica non ambigua equivalente;
- si stabilisca se il linguaggio $L(T)$ è regolare (argomentando opportunamente la risposta), e nel caso, si descriva il linguaggio attraverso 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 scope dinamico, shallow binding, assegnamento che calcola prima r-value e poi l-value, valutazione delle espressioni da sinistra a destra, valutazione degli argomenti da destra a sinistra, e indici vettori inizianti da 0, valutazione delle espressioni boolene con corto-circuito. Inoltre, per il secondo frammento di codice, mostrare l'evoluzione della CRT (non a pila nascosta).

```
int v[3]={2,1,0} , i =1;
int f(valres int[] w, ref int j){
  int i = 7;
  foreach (k : w)
    if (v[i%3] <=k  &&  j != w[i%3]++)
      write(k);
      else write(j);
  return(j);
}
write(f(v, i));
```

```
int y=2, z=4;
int F(valres int z){
  z += ++y;
  write(y, z);
  return(y+z);
};
{ int x = 0, y = 1;
  int Q (int R(valres int), ref int u){
    u = R(x);
    x = F(y);
    write(x, u);
    return(x + u);
  };
  write(Q(F, x), x, y);
};
write(y, z);
```

1. Cognome e Nome:_____ Matr.:_____

– 12 settembre 2022– A

Esercizio 4.A – Haskell (7 punti)

1. Si scrivano le seguenti funzioni Haskell:
   - una funzione che date due liste determina se una delle due è una sottolista dell'altra;
   - una funzione che data una lista determina se questa è palindroma;
   - una funzione che data una lista determina la lunghezza della più lunga sottolista finale palindroma;
   - una funzione che data una lista determina la lunghezza della più lunga sottolista palindroma.

   Si definisca il tipo di ogni funzione definita.

Esercizio 5.A – Eccezioni (2 punti)

1. (15-6-21) Date le seguenti definizione di tipi e variabili, determinare quali variabili hanno tipi equivalenti rispetto a: l'equivalenza per nome, l'equivalenza per nome lasca, l'equivalenza tra tipi di C, l'equivalenza strutturale. Si suggerisce di scrivere, per ogni relazione, le classi di equivalenza che questa determina. Una classe di equivalenza è un insieme massimale di elementi equivalenti tra loro.

```
typedef int nat;
struct pair {int[32] v; nat x};

int a, b;
nat c;
int[32] d;
nat[32] e, f;
pair g;
struct{int[32] v; int x} h;
```

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 x = 2; swap(Nat y, z){y = z, x = x}; while (x < 20) { swap(x, x) } };
```