Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
4 luglio 2022

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

1. Nella programmazione concorrente, cosa sono i canali?

2. A cosa servono le eccezioni?

3. In che modo la programmazione orientata agli oggetti permette di realizzare l'information hiding?

4. Nella gestione dello scoping dinamico, quali sono i costi nella gestione della A-list (Assiciation List)?

5. Come vengono utilizzate in C, e in altri linguaggi, le dichiarazione incomplete per definizione di funzioni mutuamente ricorsive?

6. Cos'è un blocco protetto?

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

8. Perché nella gestione dello heap esiste un problema di frammentazione?

9. Cos'è una "race condition"?

10. Quando una grammatica si dice ambigua?

11. Cosa distingue un comando da un'espressione?

12. Presentare un esempio di polimorfismo parametrico.

13. Qual'è l'utilita dei Type classes in Haskell?

14. Elencare alcune differenze la tra la programmazione funzionale e quella imperativa?

15. Cosa si intende per pre-processing di un linguaggio di programmazione?

16. Cosa si intende per memorizzazione per righe di una matrice?

17. Quali sono le caratteristiche di un weak type system?

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

1. Cognome e Nome:_____ Matr.:_____

– 4 luglio 2022– A

Esercizio 2.A – Grammatiche (4 punti)

1. Definire una grammatica, con simbolo iniziale S, che generi tutte le parole che contengono lo stesso numero di simboli a e b.

   Per la grammatica soluzione del punto precedente, o eventualmente per una grammatica alternativa, abbozzo di soluzione, rispondere alle seguenti domande:

   - si diano le stringhe di $L(S)$ di lunghezza ≤ 4;
   - si dica se la grammatica è ambigua (mostrando un testimone dell'ambiguità oppure argomentando opportunamente sulla non ambiguità);
   - si stabilisca se il linguaggio $L(S)$ è regolare (argomentando opportunamente la risposta), nel caso, lo si descriva 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 scope dinamico e deep-binding, assegnamento che calcola prima r-value e poi l-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 della CRT.

```
int v[3]={3,5,7}, i=0, j=2;
void f(name int y, valres int[] w){
  foreach(int x in  w)
    write(x += w[i++] += y);
}
f(v[--j], v);
write (v[1] + v[2]);
```

```
int x=4, y=6, z=3; int[3] v={x, y, z};
int F(ref int z){
  x += --z - y++;
  write(x, z);
  return x; }
{
  int x = -3, z = 2;
  int Q (int R(ref int), name int y){
    int q = y + x++;
    q += R(v[x+3]);
    return(q);}
  write(Q(F, F(x)));
}
write(x, y, z);
```

1. Cognome e Nome:_____ Matr.:_____

– 4 luglio 2022– A

Esercizio 4.A – Haskell (7 punti)

1. Su liste aventi come elementi coppie di valori di uno stesso tipo ordinabile, scrivere le seguenti funzioni Haskell:

   - una funzione che data una lista restituisce la lista contenente le sole coppie ordinate (il primo elemento della coppia è minore del secondo);
   - una funzione che data una lista di coppie le ordina, ossia scambia tra loro gli elementi di una coppie in modo che il primo sia minore del secondo;
   - considerando l'ordine lessicografico tra le coppie, definire una funzione che data una lista di coppie la ordina.

   Giocando sul fatto di usare o meno: le funzioni 'fold', la ricorsione di coda o le funzioni di libreria, fornire un'implentazione alternativa per ciascuna delle funzioni precedenti.

   Si definisca il tipo di ogni funzione definita.

Esercizio 5.A – Eccezioni (2 punti)

1. 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 intero;
typedef struct{int[32] v; intero x} coppia;

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

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

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

   `{ Nat x = 2; increment(Nat y) {y = y+x }; while (x < 20) { increment(x) };`

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