Cognome e Nome:_____ Matr.:_____

linguaggi di programmazione – A
15 giugno 2021

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

1. Cosa distingue un linguaggio funzionale pura da uno che non lo è?

2. Che dati sono contenuti nel record di attivazione di una procedura?

3. Cos'è una remote procedure call (RPC)?

4. A che scopo vengono utilizzati i tipi di dato astratti?

5. Quali sono le operazione tipiche sui puntatori?

6. Elencare alcune primitive per creazione di nuovi thread?

7. In un linguaggio funzionale, quali vantaggi offre la ricorsione di coda?

8. Quali sono in difetti del comando GOTO?

9. Quali sono i vantaggi dell'interpretazione rispetto alla compilazione?

10. Cos'è l'analisi sintattica?

11. Si illustri un esempio di dangling reference.

12. Cosa si intende per conversione esplicita di tipo?

13. Quali sono le caratteristiche del passaggio dei parametri per riferimento?

14. Cos'è lo Stack Pointer?

15. A cosa servono le eccezioni?

16. Per quali operazioni sono chiusi i linguaggi regolari?

17. In un linguaggi ad oggetti cosa distingue i metodi pubblici, da quelli protetti?

18. Descrive la struttura di una definizione di type class in Haskell.

1. Cognome e Nome:_____ Matr.:_____

– 15 giugno 2021– A

Esercizio 2.A – Grammatiche (4 punti)

1. Sia G la grammatica definita dalle produzioni

$$S ::= \epsilon \mid bA \mid cS$$

$$A ::= baS \mid cA$$

- si scrivano tutte le parole di lunghezza minore o uguale a 4 generate dalla grammatica,
- si stabilisca se la grammatica è ambigua,
- nel caso lo sia, si definisca una grammatica non ambigua equivalente,
- si descriva il linguaggio generato,
- si stabilisca se è un linguaggio regolare,
- nel caso, lo si descriva come espressione regolare.
- si abbozzino delle regole Lex-Alex o YACC-Happy, per la creazione di un programma che riconosca le parole generate dalla grammatica.

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-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, mostrare l'evoluzione del display.

```
char x[5] = {a,b,c,d,e};
int i=2, k=1, y[4] = {3,2,1,3};
char mess(val int i, ref char z){
  i = x[--y[i--]];
  write (y[0] + y[1] + y[3], z,  k++, i);
  return x[y[k]];
}
write(mess(++i, x[++k]));
write(x[k], y[i]);
```

```
int x = 2, y = 3;
void F (val int v , ref int r) {
  v += x ;
  r = v + y;
  write (v, r);
}
void Q (ref int y , int R( val int , ref int ) ) {
  R(y++ , y);
  y += x;
}
Q(x, F)) ;
write (x, y);
```

1. Cognome e Nome:_____ Matr.:_____

– 15 giugno 2021– A

Esercizio 4.A – Haskell (7 punti)

1. Si scrivano le seguenti funzioni Haskell:
   - una funzione che dato un naturale n costruisca la lista dei fattori primi di n, scritti in ordine crescente, es con ingresso 12 la funzione restituisce $[2, 2, 3]$
   - una funzione che date due liste di fattori primi, scritte in ordine crescente, restituisce la lista dei fattori primi presenti in almeno una delle due liste, con molteplicità massima.
   - usando le due funzione precedenti definire una funzione per il calcolo del minimo comune multiplo tra due naturali
   - si scriva una versione alternativa alla funzione precedente, usando le funzioni fold, se non usate nella versione originale, o non usandole, in caso contrario.

   Si commenti il codice e si definisca il tipo per ogni funzione definita.

Esercizio 5. – Equivalenza tra tipi (4 punti)

1. Nelle definizioni seguenti, per ogni coppia di variabili che siano equivalenti, indicare l'equivalenza più stretta che le correla. Le relazione di equivalenza da considerare sono: equivalenza per nome, equivalenza per nome lasca, equivalenza tra tipi i C, equivalenza strutturale.

```
typedef int intero;
typedef intero[32] vettore
typedef struct{vettore v; intero x} coppia
typedef coppia pair


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

Esercizio 6.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 ( x + 1) ))
   ```

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