Linguaggi di programmazione
– 15 Giugno 2022 –

1 Domande (12 punti)

... 18 domande ...

2 Grammatiche (4 punti)

Date le seguenti regole di produzione:

$$S ::= \varepsilon \mid aS \mid aSbS$$
$$P ::= \varepsilon \mid aPbP$$

siano L(S) e L(P) i linguaggi generati dalle grammatiche con simboli iniziali S e P.

- Si diano le stringhe di L(S) e L(P) di lunghezza ≤ 4;
- Si caratterizzino le stringhe di L(S) e L(P);
- Si dica se la grammatiche di S e P sono ambigue (dimostrandone l'ambiguità oppure argomentando opportunamente sulla non ambiguità);
- Si stabilisca se i linguaggi L(S) e L(P) sono regolari (argomentando opportunamente le risposte). Nel caso in cui lo siano, si descriva il linguaggio mediante un'espressione regolare.

3 Stack di Attivazione (6 punti)

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 r-value, valutazione delle espressioni da sinistra a destra, valutazione degli argomenti da sinistra a destra ed indici vettori inizianti da 0. Inoltre, per il secondo frammento di codice, si mostri anche l'evoluzione del display.

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

```
int x = 1, y = 2;
void Q(ref int r, val int v){
    v++;
    r = r + v + y;
    write(r, v);
};
void G(ref int y, int R(ref int, val int)){
    R(y, x);
    write(y);
};
G(y, Q);
write(x, y);
```

4 Haskell (7 punti)

Scrivere le seguenti funzioni in Haskell:

- una funzione che, data una lista di Boolean, conti il numero di `True`;
- una funzione che, data una lista e un valore, conti il numero di occorrenze del valore;
- una funzione che, data una lista, ritorni il numero massimo di ripetizioni (non necessariamente consecutive).

Gli elementi appartengono alla `type class` Eq e non a Ord.
Per le implementazioni usare fold, ricorsione di cosa o le funzioni precedentemente definite. Fornire un'implementazione alternativa per ciascuna.
Si definisca il tipo di ogni funzione definita. Si commenti il codice.

5 Eccezioni (2 punti)

Descrivere il comportamento del programma con uno dei seguenti meccanismi di passaggio parametri: valore, riferimento, valore-risultato, nome.

```
{ int y=2;
    void f(___ int x) throws E(){
        x = x + y;
        throw new E();
        x = x + y;
    }
    try{f(y)} catch(E) {y++};
    write(y);
}
```

6 Sistema di assegnazione di tipo (2 punti)

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

```
\z : Bool*Nat, if(first z) then(second z) + 1 else(second z) - 1
```

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