Esame di LP1

Prof Piero Bonatti

22 Gennaio 2015

Domande generali – Max 6 punti

Esercizio 1: **[2 punti]** Barrare tutte le frasi vere.

a) [ ] SQL è un linguaggio di programmazione general purpose.
b) [ ] Nel paradigma funzionale puro non c'è memoria, solo ambiente.
c) [ ] Nel paradigma funzionale puro non esistono cicli, ma solo ricorsione.
d) [ ] Nel paradigma logico non c'è distinzione netta tra input e output.
e) [ ] Il primo FORTRAN non ha gestione dinamica della memoria.

Esercizio 2: **[2 punti]** Dato il codice C:

```
int x, *y;
y = &x;
```

Barrare l'interpretazione corretta della espressione &x nell'assegnamento:

[ ] *env*(x) [ ] *mem*(*env*(x)) [ ] *env*(*mem*(x)) [ ] *mem*(*mem*(*env*(x)))

Esercizio 3: **[2 punti] (Barrare tutte le riposte corrette)** Date le dichiarazioni in C:

```
typedef struct {int a;} Type1;
typedef struct {int a;} Type2;
Type1 v1;
Type2 v2;
```

Dire se l'assegnamento:

```
v1 = v2;
```

[ ] rispetta la name equivalence [ ] rispetta la structural equivalence [ ] è corretto in C

Esercizio sul passaggio di parametri – Max 11 punti

Dire qual è l'output del seguente programma nei casi elencati qui sotto:

1. Scoping dinamico, [MODE] = IN per copia
2. Scoping dinamico, [MODE] = IN per riferimento
3. Scoping statico, [MODE] = IN per copia
4. Scoping statico, [MODE] = IN OUT per riferimento

Mostrare gli stack di attivazione (pena la perdita di punti), tranne nei casi di errore, nei quali bisogna invece indicare l'istruzione che causa l'errore.

```
program p1
int q; int r; int s; int t;
     procedure p2([IN OUT x rif] int q,[MODE] int r)
     int s;
          procedure p3([IN OUT x rif] int q)
          int t; int r;
          BEGIN
          t=q;
          r=q-2;
          q=r+4;
          s=4;
          write(q,r,s,t);
          END

          procedure p4([IN OUT x copia] int s)
          int q;
          BEGIN
          q=t;
          s=q;
          r=4;
          t=t;
          p3(q);
          write(q,r,s,t);
          END

     BEGIN
     s=t+3;
     q=3;
     r=3;
     t=2;
     p4(q);
     write(q,r,s,t);
     END

BEGIN
q=2;
r=4;
s=4;
t=3;
p2(r, s);
write(q,r,s,t);
END
```

UML – Max 15 punti

Progettare un sistema di supporto al lavoro di ufficio come descritto in seguito. Vi sono dei compiti, ciascuno dei quali ha una descrizione testuale, un responsabile (caratterizzato da nome, telefono e email) e una data entro cui il compito deve essere svolto. La data è rappresentata con un intero. Un compito può essere elementare o composito. Quelli compositi rappresentano una sequenza di compiti più semplici; per questo posseggono anche una lista ordinata di compiti (che possono essere elementari o compositi a loro volta). I compiti supportano una operazione di notifica: se la data odierna è maggiore della data di scadenza del compito, allora viene mandata una mail al responsabile per avvisarlo. Il componente che fornisce la data odierna e quello che invia la mail non sono sotto la vostra responsabilità (li sviluppa qualcun altro). Sapete solo che il primo supporta un metodo *get_date*() che restituisce la data, e che il secondo supporta un metodo *send_mail*(...) i cui input sono il destinatario e il testo del messaggio.

Esercizio 1: Disegnare un diagramma delle classi per queste specifiche.
[max 8 punti]

Esercizio 2: Disegnare un diagramma delle sequenze per lo scenario seguente, mostrando tutte le interazioni tra gli oggetti coinvolti. **Se un oggetto ha bisogno di un attributo privato di un altro oggetto dovete mostrare come se lo procura**.

L'utente invoca il metodo di notifica sul compito elementare C0, che ha scadenza D0. La data odierna è D > D0. Il metodo di notifica legge la data odierna mediante l'apposito componente; poichè è maggiore di D0, il metodo di notifica invia la mail al responsabile di C0 mediante l'apposito componente.

[max 7 punti]