Fondamenti di Informatica T-1 (A.A. 2008/2009) - Ingegneria Informatica
Prof.ssa Mello
Prova d'Esame di Giovedì 17 Settembre 2009 – durata 1h
Totale 12 punti, sufficienza con 7

ESERCIZIO 1 (5 punti)

Un sistema di gestione delle presenze sul luogo lavorativo assegna ad ogni impiegato un codice (un intero). Ogni impiegato all'ingresso nel luogo di lavoro "timbra il cartellino", e così ad ogni uscita. Il sistema tiene traccia in due liste differenti le timbrature in ingresso e le timbrature in uscita: ogni volta che un impiegato entra/esce, il sistema aggiunge il codice identificativo in testa alla lista opportuna; uno stesso codice compare quindi ripetuto più volte. In un mondo perfetto il numero di ingressi sul luogo di lavoro ed il numero di uscite dovrebbe essere uguale. Invece ogni giorno ci sono dei problemi.
Si scriva una funzione ricorsiva:

```
list estrai(list l1, list l2, list l3);
```

che riceve in ingresso:

- una lista **`l1`** di interi, rappresentanti i codici identificativi di tutti gli impiegati (codici non ripetuti);
- una lista **`l2`** che contiene i codici degli impiegati che nella giornata odierna hanno timbrato l'ingresso (ad ogni timbro corrisponde un codice: più timbri di ingresso effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);
- una lista **`l3`** che contiene i codici degli impiegati che nella giornata odierna hanno timbrato l'uscita (ad ogni timbro corrisponde un codice: più timbri di uscita effettuati dalla stessa persona corrispondono allo stesso codice che compare più volte ripetuto nella lista);

La funzione **`estrai(…)`** deve restituire in uscita una lista dei codici corrispondenti alle persone che effettivamente son venuti in ufficio, e che hanno un numero di entrate/uscite discrepante.
Ad esempio, supponendo **`l1=[3,7,9]`**, **`l2=[7,7,9,7,9]`**, **`l3=[7,9,9,9,7,7]`**, la lista restituita come risultato sarà **`[9]`**, poiché l'impiegato identificato dal codice **`3`** non è venuto al lavoro (non è presente né in **`l2`**, né in **`l3`**): l'impiegato identificato dal codice **`7`** è a posto (tre ingressi in **`l2`**, tre uscite in **`l3`**); invece l'impiegato identificato dal codice **`9`** ha due ingressi e tre uscite.

Al fine di realizzare la funzione di cui sopra, il candidato implementi una funzione iterativa:

```
int conta(element el, list l1);
```

che restituisce il numero di volte che l'elemento **`el`** è presente nella lista **`l1`**.

ESERCIZIO 2 (3 punti)

Dati i numeri decimali **`A=62`**, e **`B=-5`**, si determini il numero minimo di bit necessari per rappresentare contemporaneamente i suddetti numeri in complemento a 2. Utilizzando poi lo stesso numero di bit, si esegua l'operazione:

```
C = A-B
```

e si discuta se il risultato ottenuto è o no significativo, fornendo una opportuna spiegazione.

ESERCIZIO 3 (2 punti)

Si introduca brevemente la struttura dati dinamica "stack" vista a lezione, descrivendone le funzioni primitive di accesso e la loro implementazione.

ESERCIZIO 4 (2 punti)

Data la funzione:

```
float g(float a, float b){
      if ( (a==b) || (a<=0))
            return a;
      else
            return 1 + g(a-2, b) + g(a-1, b-0.5);
}
```

mostrare la sequenza dei record di attivazione nel caso in cui la funzione sia invocata con parametri attuali (2.0, 1.0) e il valore di ritorno.

Soluzioni

ESERCIZIO 1

```
int conta(element el, list l1) {
      int result = 0;
      while (!empty(l1)) {
            if (el==head(l1))
                  result++;
            l1=tail(l1);
      }
      return result;
}


list estrai(list l1, list l2, list l3) {
      if (empty(l1))
            return l1;
      else {
            if (conta(head(l1), l2) != conta(head(l1), l3))
                  return cons(head(l1), estrai(tail(l1), l2, l3));
            else return estrai(tail(l1), l2, l3);
      }
}
```

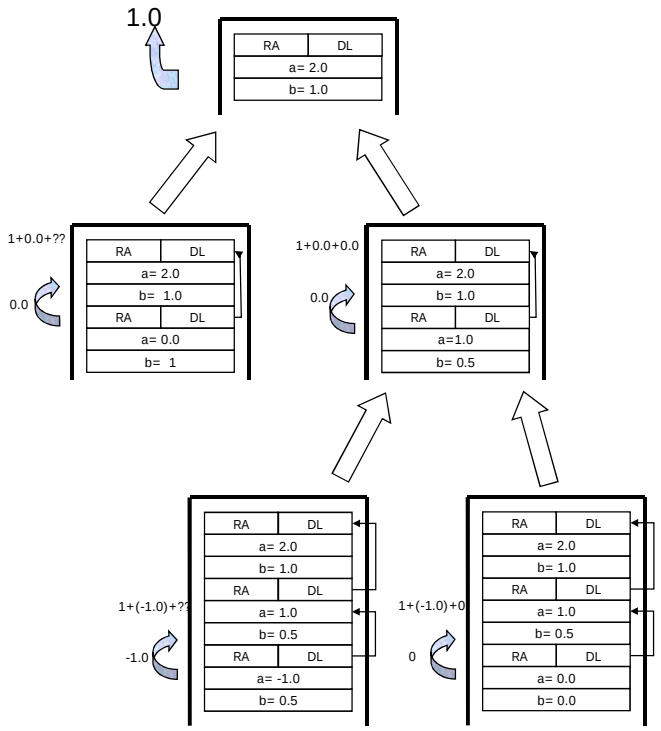
ESERCIZIO 2

Il maggiore dei due numeri, in valore assoluto, è A, che vale 62. Per rappresentare il valore assoluto di A sono sufficienti 6 bit (2^6=64). Al fine di rappresentare anche i numeri negativi, tramite la notazione a complemento a due, è necessario utilizzare nella rappresentazione un ulteriore bit. Perciò il numero minimo di bit, necessari e sufficienti per rappresentare A e B è 7.

```
 62 ->     0111110                  0111110 ( 62) -
  5 ->     0000101                  1111011 ( -5) =
           1111010                  ----------
 -5 ->     1111011                  1000011 (???)
```

Facendo il conto in base dieci, il risultato atteso è 67. Eseguendo il calcolo in base 2 invece il risultato conseguito è 1000011, che convertito in base dieci (secondo notazione a complemento a due) risulta essere pari a -61. Tale errore è ovviamente dovuto ad un problema di rappresentazione: il risultato dell'operazione non può essere rappresentato con soli 7 bit.
NOTA: In questa soluzione l'operazione aritmetica di sottrazione è stata effettuata tramite l'algoritmo della sottrazione. Molte architetture hardware sfruttano le caratteristiche del complemento a 2 ed eseguono solo gli algoritmi di addizione. La sottrazione di questo esercizio poteva quindi anche essere eseguita seguendo tale modello hardware: si effettua di nuovo il complemento della rappresentazione di (-5), e poi si esegue una addizione.

ESERCIZIO 4


1.0
RA
DL
a= 2.0
b= 1.0
1+0.0+??
0.0
RA
DL
a= 2.0
b= 1.0
RA
DL
a= 0.0
b= 1
1+0.0+0.0
0.0
RA
DL
a= 2.0
b= 1.0
RA
DL
a=1.0
b= 0.5
1+(-1.0)+??
-1.0
RA
DL
a= 2.0
b= 1.0
RA
DL
a= 1.0
b= 0.5
RA
DL
a= -1.0
b= 0.5
1+(-1.0)+0
0
RA
DL
a= 2.0
b= 1.0
RA
DL
a= 1.0
b= 0.5
RA
DL
a= 0.0
b= 0.0