Fondamenti di Informatica T-1 (A.A. 2008/2009) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Venerdì 13 Febbraio 2009 – durata 1h
Totale 12 punti, sufficienza con 7 - Compito A

ESERCIZIO 1 (4 punti)

Si scriva una funzione iterativa:

```
int isAdd(list l1, list l2, list add);
```

che ricevute in ingresso tre liste di interi, determini se la lista **add** contiene in ogni elemento la somma degli elementi corrispondenti della lista **l1** e la lista **l2**. In particolare la funzione deve restituire un valore maggiore di 0 se effettivamente vale questa proprietà; deve ritornare 0 altrimenti. Si supponga per semplicità che le liste **l1** e **l2** abbiano la stessa lunghezza. Non si può supporre invece questo per la lista add.
Ad esempio, date le liste **l1=[1,2,3,4]**, **l2=[1,4, 3, 0]**, **add=[2,6,6,4]**, la funzione deve restituire valore 1. Invece, se invocata con **l1=[1,2,3,4]**, **l2=[1,4,5,7]**, **add=[3,2,0,1]**, oppure **add=[2,6,8]** o **add=[2,6,8,11,20]** allora la funzione deve restituire 0.
La funzione deve essere realizzata utilizzando il tipo di dato astratto **list**, definito per gli interi (non è necessario riportare la definizione nella soluzione). Si possono utilizzare le sole operazioni primitive definite durante il corso, che quindi possono NON essere riportate nella soluzione. Non si possono usare altre funzioni di alto livello.

ESERCIZIO 2 (3 punti)

Data la funzione:

```
int p1(int x, int y) {
      float a = x;

      if (a<y)
            return a / 2 + 0.5;
      else
            return p1( a, y+1) + p1( a, y+1.5);
}
```

mostrare la sequenza dei record di attivazione nel caso in cui la funzione sia invocata con parametri attuali (4, 3) e il valore di ritorno. Si ignorino eventuali warning che potrebbero essere generati in fase di compilazione.

ESERCIZIO 3 (3 punti)

Si introduca il metodo di ordinamento di un vettore “a bolle” (bubble sort) e poi si scriva lo (pseudo) codice di una funzione che lo realizza.

ESERCIZIO 4 (2 punti)

Si consideri il seguente codice:

```
void cambia(int v[][2]) {
      v[1][1] = 10;
}

int main(void)
{
      int a[2][3] = {{1,2,3}, {4,5,6}};
      int i, j;

      cambia(a);
      for (i=0; i<2; i++) {
            for (j=0; j<3; j++)
                  printf("%d ", a[i][j]);
            printf("\n");
      }     return (0);
}
```

Qualora il compilatore compili questo programma, quali sono i valori stampati a video a tempo di esecuzione? (si motivi la risposta in maniera opportuna)
Si spieghi inoltre perchè nel passaggio di una matrice come parametro di una funzione, non è significativa la prima dimensione mentre la seconda (nel caso la matrice sia bi-dimensionale) è invece necessaria

Soluzioni

ESERCIZIO 1

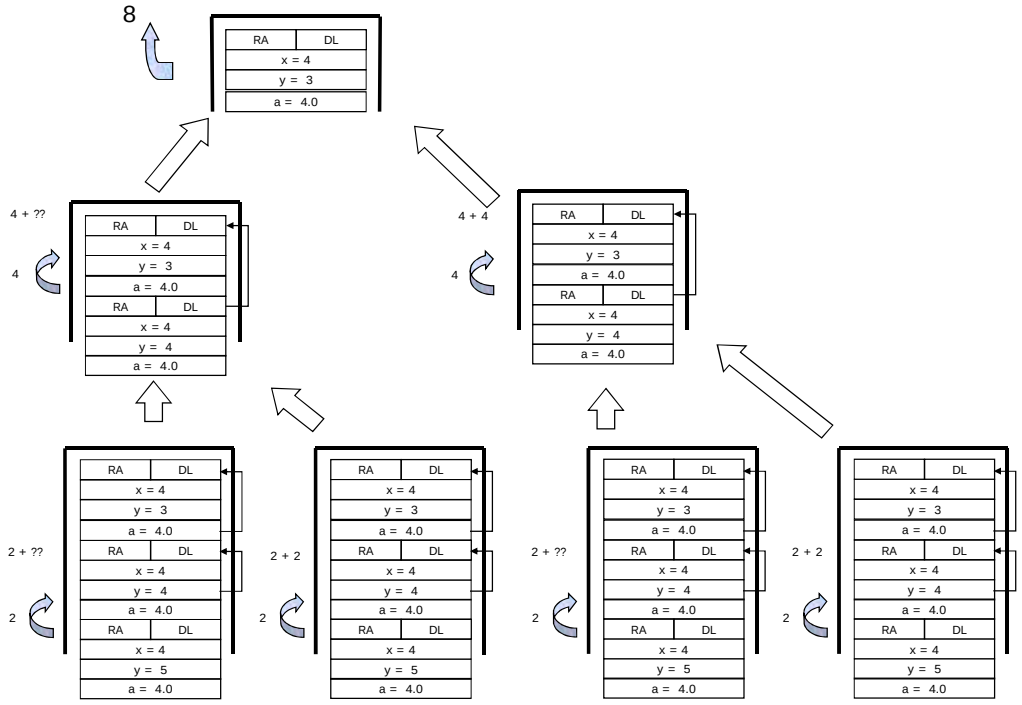
```
int isAdd(list l1, list l2, list add) {
      int problem = 0;

      while (!empty(l1) && !empty(add) &&  !problem) {
            if ((head(l1) + head(l2))== head(add) ) {
                  l1 = tail(l1);
                  l2 = tail(l2);
                  add = tail(add);

            }

                  else problem=1;
      }
      if (!problem && empty(l1) && empty(add) ) return 1;
      else return 0;
}
```

ESERCIZIO 2



ESERCIZIO 4

Il programma è corretto, compila correttamente e stampa:

```
1 2 3
10 5 6
```

Si noti che, siccome nella dichiarazione della funzione la seconda dimensione dell'array passato come argomento è differente dalla seconda dimensione dell'array (stando a come è stato dichiarato nel main), la cella [1][1] riferita nella funzione è differente dalla cella [1][1] riferita nel main.