Analisi e progettazione del software
Compito di metà corso
26 novembre 2012

Esercizio 1 (punti 8) Si consideri la classe `Pila` (di interi) vista a lezione. Si scriva una funzione esterna (non *friend*) booleana che riceva come parametro un oggetto di tipo `Pila` e verifichi se i suoi elementi sono ordinati in senso strettamente crescente partendo dal fondo fino all'elemento affiorante. Ovviamente, una pila vuota o con un solo elemento è sempre ordinata.

Esercizio 2 (punti 10) Un file contiene una sequenza di appuntamenti, uno per riga, formati dal nome (stringa senza spazi) e dalla data.

Come esempio, si consideri il seguente file

```
Dentista 4/12/2012
Partita_di_Tennis 27/11/2012
Lezione 18/11/2012
Compitino 26/11/2012
Discoteca 19/11/2012
```

Si scriva una funzione che riceva come parametro il nome di un file siffatto e *modifichi il file stesso* eliminando gli appuntamenti *passati*, cioè quelli precedenti alla data odierna.

A questo scopo si assuma che il costruttore senza parametri della classe `Data` assegni all'oggetto la data corrente di sistema. Si assuma inoltre che la classe `Data` dia dotata di tutti gli operatori e le funzioni che si ritengono opportuni.

Inoltre, la funzione deve restituire l'*orizzonte di pianificazione*, cioè il numero di giorni dalla data odierna a quella dell'appuntamento più avanti nel tempo.

Nel file di esempio e con data odierna 26/11/2012, la funzione restituisce il valore 8 (il `Dentista` è l'appuntamento più avanti ed è tra 8 giorni) e il file modificato sarà il seguente

```
Dentista 4/12/2012
Partita_di_Tennis 27/11/2012
Compitino 26/11/2012
```

Esercizio 3 (punti 14) Si considerino le classi `A` e `B` definite qui sotto

```
class B
{
 public:
  B(int mx) { x = mx; }
  void SetX(int mx) { x = mx; }
  int X() const { return x; }
 private:
  int x;
};

class A
{
 public:
  A(unsigned n, int e);
  int operator[](unsigned i) const
     { return v1[i].X() + v2[i]; }
  void Set(unsigned i, int e = 0);
 private:
  vector<B> v1;
  int* v2;
};
```

```
A::A(unsigned n, int e)
  : v1(n,e)
{
  v2 = new int[n];
  for (unsigned i = 0; i < n; i++)
    v2[i] = e;
}

void A::Set(unsigned i, int e)
{
  assert(i < v1.size());
  v1[i].SetX(e);
  v2[i] = e;
}
```

e la seguente funzione `main()`.

```
int main()
{
  A a1(4,8), a2(3,5), a3(10);
  a2.Set(2);
  a2.Set(1,3);
  a1[0] = 5;
  a1.Set(5,2);
  cout << a1 << endl;
  cout << a2[0] << " " << a2[1] << " " << a2[2] << endl;
  return 0;
}
```

3.1 (3 punti) Segnalare le istruzioni (o parti di esse) della funzione `main()` che danno errore in compilazione o in esecuzione e spiegare brevemente il motivo.

3.2 (3 punti) Riportare cosa stampa il programma una volta eliminate le parti che generano errore.

3.3 (8 punti) Scrivere il costruttore di copia, l'operatore di assegnazione e il distruttore della classe `A` (e, se necessario, della classe `B`) in modo che evitino la condivisione di memoria tra oggetti della classe `A`.