UNIVERSITÀDELLACALABRIA

Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Informatica
Prova di Programmazione ad Oggetti Data: 6 Luglio 2012

Esercizio 1

Una società di poste private ci ha chiesto di realizzare un sistema per la registrazione e la lavorazione dei vari prodotti postali. Il software in questione verrà utilizzato dagli impiegati dell'azienda per caricare i vari prodotti, questi sono gestiti in ordine di priorità e poi in ordine di arrivo utilizzando la classe coda di cui riportiamo una specifica parziale all'interno del riquadro. I prodotti estratti dalla coda sono elaborati dal sistema (vedi la procedura *elabora()* nel riquadro) effettuando una lavorazione specifica per ogni prodotto postale (lettera, raccomandata, assicurata, posta prioritaria, pacco, pacco raccomandato, ecc. ecc.) dipendente dalla classe di appartenenza (richiamando il metodo *eseguiLavorazione(…)*). Un requisito fondamentale è che il sistema sia progettato in modo da essere flessibile rispetto all'aggiunta o alla rimozione di prodotti postali e, in particolare, la classe coda non deve subire alcuna modifica in caso di aggiunta/rimozione di prodotti.

```
class CodaProdottoPostale :
      protected list<ProdottoPostale*> {
public:
 CodaProdottoPostale();
 ProdottoPostale* pop();
 unsigned size() const;
 ProdottoPostale* top();
 void push(ProdottoPostale* p){
   list<ProdottoPostale*> it = begin();
   while ( it != end() && *p >= *(*it) )
      it++;
   insert(it,p);
 }
};
…
void SistemaPostale::elabora(){
   while (coda.size() > 0){
     ProdottoPostale* p = coda.pop();
     p->eseguiLavorazione(this);
     delete p;
   }
}
```

A tale scopo si richiede:

(a) di progettare le interfacce (solo file intestazione) delle classi che modellano i prodotti postali utilizzando l'ereditarietà e il polimorfismo;

(b) implementare i metodi lasciati incompleti nella classe coda riportata nel riquadro.

Nota: è consentito l'uso di STL; è possibile aggiungere classi di supporto fino ad un massimo di 4 identificatori di tipo oltre quelli già utilizzati nel codice riportato nel riquadro.

Esercizio 2

Discutere la complessità asintotica dell'algoritmo implementato dalla seguente procedura che riceve in input un vettore ordinato di numeri interi ed un numero intero di cui restituisce la posizione all'interno del vettore (-1 se l'elemento non occorre nel vettore).

```
int find(int x, vector<int> v){
      int f = 0;  int t = v.size();
      if((t-f) == 0) return -1;
      do {
            int half = f + (t-f)/2;
            if(v[half] == x) return half;
            if(v[half] > x) t = half;
            else f = half+1;
      } while(f <= t);
      return -1;
}
```