```
1: class Crociera {
2: public:
3:   Crociera(Nave* n){ this->nave = n; }
4:   Crociera(const Crociera& c);
5:   ~Crociera();

6:   //aggiunge la persona se non è presente e se non sfora la capienza della nave
7:   bool aggiungiPasseggero(Persona*);

8:   //rimuove una persona se è presente (mantenendo l'ordine di inserimento)
9:   bool rimuoviPasseggero(Persona*);

10:  //restituisce il numero di passeggeri
11:  unsigned int numPasseggeri() const;

12:  //stampa i dati di tutti i passeggeri e della nave
13:  friend ostream& operator<<( ostream& out, const Crociera& c);

14:  unsigned int m1() const {
15:   int count = 0;
16:   for(int i = 0, j = 0; (2*i+j) < passeggeri.size(); i++, j--) {
17:    count++;
18:   }
19:   if(count != passeggeri.size()) {
20:    for(int k = 0; k < passeggeri.size(); k++) {
21:      for(int t = 0; t < passeggeri.size(); t++) {
22:       count += t*k;
23:      }
24:    }
25:   }
26:   return count;
27:  }

28: protected:
29:   virtual double calcolaPrezzoTotale() const = 0;

30: private:
31:   Nave* nave;
32:   vector<Persona*> passeggeri;
33: };

34: class C1 : public Crociera {
35: public:
36:   C1(Nave* n) : Crociera(n) {}
37:   double calcolaPrezzoTotale() const { return 500.0 * numPasseggeri(); }
38: };

39: class C2 : protected Crociera {
40: public:
41:   C2(Nave* n) : Crociera(n) {}
42:   double calcolaPrezzoTotale() const { return 1000.0 * numPasseggeri(); }
43: };
```

Programmazione Ad Oggetti. 09 Febbraio 2016

Cognome: _____ Nome: _____

Matricola: _____ Esercizi lab: _____

Data la classe Crociera:

- Implementare i metodi alle linee 7, 9, 11, 13.
- Riportare una possibile definizione della classe Nave (solo Nave.h).
- Discutere la complessità del metodo m1().

Rispondere alle seguenti domande a scelta multipla:

1. Quale tra queste implementazioni del distruttore di Crociera è la più indicata?
   a) `~Crociera{ delete nave; }`
   b) `~Crociera{ passeggeri.clear(); }`
   c) `~Crociera{ delete nave; delete [] passeggeri; }`
   d)
   ```
   ~Crociera{
     for(int i = 0; i < passeggeri.size(); i++)
       delete passeggeri[i];
   }
   ```
2. Quale tra queste implementazioni del costruttore di copia di Crociera è la più indicata?
   a) `Crociera(const Crociera& c){ passeggeri = c.passeggeri; nave = c.nave; }`
   b)
   ```
   Crociera(const Crociera& c){
       passeggeri = c.passeggeri; delete nave;
       nave = new Nave[c.numPasseggeri()]; }
   ```
   c)
   ```
   Crociera(const Crociera& c){
       nave = new Nave(); nave->setNumeroPasseggeri(c.nave); }
   ```
   d) `Crociera(const Crociera& c){ passeggeri = c.passeggeri; }`
3. Quale tra le seguenti operazioni NON è consentita nel file main.cpp
   a) `Crociera c* = new C1(n);  c->calcolaPrezzoTotale();`
   b) `Crociera c* = new C2(n);  c->calcolaPrezzoTotale();`
   c) `Sono entrambe consentite`
   d) `Nessuna delle due è consentita`
4. Quale tra le seguenti operazioni NON è consentita nel file main.cpp
   a) `Crociera c* = new C1(n);  c->numPasseggeri();`
   b) `Crociera c* = new C2(n);  c->numPasseggeri();`
   c) `Sono entrambe consentite`
   d) `Nessuna delle due è consentita`