Esercizio 1.

```
// costruttore di copia della classe Taxi
Taxi::Taxi( const Taxi& aTaxi )
{
  codice = aTaxi.codice;
  nomeGuidatore = aTaxi.nomeGuidatore;
  occupato = aTaxi.occupato;
  posizioneCorrente = aTaxi.posizioneCorrente;
}


// metodo occupato() della classe Taxi
bool Taxi::occupato()
{
  return occupato;
}


// operatore << della classe Taxi
ostream& Taxi::operator<<( ostream& out, const Taxi& taxi )
{
  out << "TAXI" << endl;
      << "  Codice: " << taxi.codice << endl
      << "  Nome Guidatore: " << taxi.nomeGuidatore << endl
      << "  Stato: " << ( taxi.occupato ? "occupato" : "libero" ) << endl
      << "  Posizione: " << taxi.posizioneCorrente << endl;

  return out;
}


// operatore = della classe VettoreTaxi
VettoreTaxi& VettoreTaxi::operator=( const VettoreTaxi& right )
{
  this.codice = right.codice;
  this.nomeGuidatore = right.nomeGuidatore;
  this.occupato = right.occupato;
  this.posizioneCorrente = right.posizioneCorrente;
}


// costruttore della classe VettoreTaxi
VettoreTaxi::VettoreTaxi( int l )
{
  assert( l >= 0 );

  lunghezza = l;
  elenco = new Taxi[ lunghezza ];
}



// distruttore della classe VettoreTaxi
VettoreTaxi::~VettoreTaxi()
{
  delete [] elenco;
}
```

Esercizio 2.

```
ListaCliente clientiAbitualiGold(
    const ListaCliente& lclienti,
    const ListaBiglietto& lbiglietti )
{
  ListaCliente gold;

  IteratoreCliente itC( lclienti );
  while( ! itC.isNull() )
  {
    int numeroBiglietti = 0;
    int spesaTotale = 0;

    IteratoreBiglietto itB( lbiglietti );
    while( ! itB.isNull() )
    {
      if( itB.getCurrentValue().codiceCliente == itC.getCurrentValue().codiceCliente )
      {
        numeroBiglietti++;
        spesaTotale += itB.getCurrentValue().prezzo;
      }
    }

    if( numeroBiglietti >= 100
        && ( spesaTotale / numeroBiglietti >= 100 ) )
      gold.addInTesta( itC.getCurrentValue() );
  }

  return gold;
}
```

Esercizio 3.

```
class ClienteAbbonato : public Cliente
{
public:
  int codiceAbbonamento;
  ListaMyString eventiCopertiAbbonamento;
  int annoValiditaAbbonamento;


  // costruttore di default
  ClienteAbbonato();

  // costruttore con parametri
  ClienteAbbonato(
      int aCodiceCliente,
      MyString aNome,
      int aCodiceAbbonamento,
      ListaMyString& aEventiCopertiAbbonamento,
      int aAnnoValiditaAbbonamento );

  // costruttore di copia
  ClienteAbbonato( const ClienteAbbonato& );
}

ClienteAbbonato(
    int aCodiceCliente,
    MyString aNome,
    int aCodiceAbbonamento,
    ListaMyString& aEventiCopertiAbbonamento,
    int aAnnoValiditaAbbonamento )
{
  codiceCliente = aCodiceCliente;
  nome = aNome;
  codiceAbbonamento = aCodiceAbbonamento;
  eventiCopertiAbbonamento = aEventiCopertiAbbonamento;
  annoValiditaAbbonamento = aAnnoValiditaAbbonamento;
}
```