Esempio esercizio esame (appello 20050222):

La classe templatica Set è intesa rappresentare un insieme di elementi di tipo T. L'implementazione della classe si basa sulla manipolazione di liste ordinate (senza duplicati). L'interfaccia della classe presenta numerosi problemi.

Cercare di individuarne il maggior numero ed indicare come possono essere risolti (riscrivendo l'interfaccia).

```
template
class Set : public std::list {
public:
 // Costruisce l'insieme vuoto.
 Set();
 // Costruisce il singoletto che contiene t.
 Set(T t);
 Set(Set y);
 void operator=(Set y);
 virtual ~Set();

 unsigned int size();
 bool is_empty();
 bool contains(Set y);

 T& min();
 void pop_min();
 void insert(T z);
 void union_assign(Set y);
 void intersection_assign(Set y);

 void swap(Set y);
 std::ostream operator<<(std::ostream os);

private:
// ...
};
```

Risoluzione:

```
template
class Set {
private:
 std::list m_list; // m: sta per dato membro
 // In questo modo garantisco l'invariante di classe
public:
 // Costruisce l'insieme vuoto.
 Set();

 // Costruisce il singoletto che contiene t.
 // explicit è per evitare la conversione implicita
 explicit Set(const T& t);

 // Se lo avessi lasciato senza il const sarebbe andato in un loop
 // chiamando se stesso all'infinito
 Set(const Set& y);

 // se non voglio permettere la concatenazione di assegnamenti
 // il tipo di ritorno della funzione deve essere void, in caso contrario
 // il tipo deve essere Set& (commento)
 void operator=(const Set& y);
 // Set& operator=(const Set& y);

 // Distruttore
 virtual ~Set();

 // I tre metodi seguenti non vanno a modificare i dati privati, quindi
 // si seguire la "regola" del const correctness

 unsigned int size() const;
 // I due metodi successivi sono definiti predicati
 bool is_empty() const;
 bool contains(const Set& y) const;

 // Generalmente il metodo che segue viene reso disponibile in
 // sola lettura e, se non si rischia di violare l'invariante di classe,
 // in lettura e in scrittura
```

```
 const T& min() const; // L'insieme restituito non è modificabile

 // T& min(); // L'insieme restituito è modificabile, quindi
 // potrebbe rendere compromissibile l'invariante di classe

 void pop_min();
 void insert(const T& z);
 void union_assign(const Set& y);
 void intersection_assign(const Set& y);

 void swap(Set&& y);

 // Se il metodo seguente fosse implementato all'interno della classe,
 // comporterebbe una chiamata del tipo s << std::cout;
 // così violando il principio di sorpresa minima
 // std::ostream& operator<<(std::ostream& os) const;
};

template
std::ostream& operator<<(std::ostream& os, const Set& t);
```