Esercizio 1 - Classe base e operatori

Definire una classe ComplexNumber i cui oggetti rappresentano numeri complessi nella forma a + bi. La classe deve includere:

1. Due campi dati private per memorizzare parte reale e immaginaria
2. Costruttori appropriati
3. Overloading degli operatori di:
    - somma tra ComplexNumber
    - moltiplicazione tra ComplexNumber
    - conversione esplicita a double (che restituisce il modulo del numero complesso)
4. Un metodo conjugate() che restituisce il complesso coniugato
5. Overloading dell'operatore di output

Scrivere un programma che testi tutte le funzionalità implementate.

Esercizio 2 - Gestione della memoria

```
#include<iostream>
using namespace std;

class Array {
private:
    int* data;
    int size;
public:
    Array(int s = 5): size(s) {cout << "C1 "; data = new int[size];}
    Array(const Array& a): size(a.size) {
        cout << "C2 ";
        data = new int[size];
        for(int i=0; i<size; i++) data[i] = a.data[i];
    }
    ~Array() {cout << "D "; delete[] data;}
    void set(int i, int v) {if(i<size) data[i]=v;}
};

class Container {
private:
    Array arr;
    Array* ptr;
public:
```

```
    Container(): ptr(new Array()) {cout << "CC1 ";}
    Container(const Container& c): arr(c.arr), ptr(new Array(*c.ptr)) {cout <<
"CC2 ";}
    ~Container() {cout << "DC "; delete ptr;}
};
```

Cosa stampa il seguente main()?

```
int main() {
    Container c1;
    cout << "UNO\n";
    Container c2(c1);
    cout << "DUE\n";
    return 0;
}
```

Esercizio 3 - Template e gestione delle eccezioni

Implementare un template di classe Stack<\T> che:

1. Gestisce uno stack di elementi di tipo T
2. Ha un limite massimo di elementi configurabile nel costruttore
3. Implementa i metodi:
    - push() che lancia un'eccezione StackOverflow se lo stack è pieno
    - pop() che lancia un'eccezione StackUnderflow se lo stack è vuoto
    - top() che restituisce l'elemento in cima senza rimuoverlo
    - isEmpty() e isFull() per verificare lo stato dello stack
4. Definire le classi di eccezione StackOverflow e StackUnderflow
5. Includere un costruttore di copia profonda e l'operatore di assegnazione

Sì, posso creare altri esercizi simili a quelli delle prime pagine del pdf:

Esercizio 4 - Tipo astratto con campo statico

Definire una classe Counter i cui oggetti rappresentano contatori incrementali con reset. Il valore massimo raggiungibile dai contatori deve essere dichiarato come campo dati statico. Definire metodi statici di set_max() e get_max() per tale campo statico.
Devono essere disponibili gli operatori di:

- incremento prefisso e postfisso
- conversione al tipo primitivo int

- reset() che riporta il contatore a 0
  Definire inoltre opportuni convertitori di tipo affinché questa classe sia liberamente utilizzabile assieme al tipo primitivo int.
  Scrivere un programma d'esempio che utilizza tutti i metodi della classe.

Esercizio 5

Il seguente programma compila. Quali stampe produce la sua esecuzione?

```
#include<iostream>
using std::cout;

class Box {
private:
    int val;
public:
    Box(int k = 3): val(k) {cout << k << " B01 ";}
    Box(const Box& b): val(b.val) {cout << "Bc ";}
    Box f() const {return *this;}
};

class Container {
private:
    Box boxes[2];
    static Box b;
public:
    Container() {boxes[1] = Box(5); cout << "C0 ";}
    Container(const Container& c) {cout << "Cc ";}
};

Box Container::b = Box(7);

Box Fun(Box* p, const Box& b, Container c) {
    *p = b;
    b.f();
    return *p;
}

main() {
    cout << "ZERO\n";
    Box b1; cout << "UNO\n";
    Box b2(4); cout << "DUE\n";
    Box* p = &b1; cout << "TRE\n";
    Container c; cout << "QUATTRO\n";
    b1 = Fun(p,b2,c); cout << "CINQUE\n";
```

```
    Box b3 = Fun(&b1,*p,c); cout << "SEI";
}
```

Esercizio 6

Perché il seguente programma non compila? Modificare o eliminare una e soltanto una delle righe per far compilare il programma.

```
class Node {
public: // 1
    const int* next; // 2
    Node(int n=0): next(new int(n)) // 3
    { } // 4
};

main() {
    Node x(3); // 5
    Node y; // 6
    x = y; // 7
    Node z(y); // 8
}
```