Esercizio 1

Dobbiamo definire una classe ***intmod*** i cui oggetti rappresentano interi modulo *m*. Dobbiamo definire un campo modulo statico, un metodo *get_modulo* e *set_modulo* per settare il modulo. Dobbiamo ridefinire gli operatori +, * , ++ e definire opportune conversioni in modo che ogni volta che si ha un intero e un oggetto di tipo *intmod*.

intmod im;
int i;
i + im // deve restituire un risultato intero

intmod.h

```
#ifndef INTMOD
#define INTMOD

class intmod{

public:
    explicit intmod (int n=0);  // Il costruttore di conversione viene invocato solo esplicitamente

    operator int() const;
    static int get_modulo();
    static void set_modulo(int mod);
    intmod operator+(intmod im) const;
    intmod operator*(intmod im) const;
    intmod operator++();    //PREFISSO
    intmod operator++(int);     //POSTFISSO

private:
    static int modulo;
    int val;
};

int intmod::modulo=5;   //Inizializzazione di un campo dati statico
```

intmod.cpp

```
#include "intmod.h"

intmod::intmod(cnt n)
    : val = n%modulo {}

intmod::operator int() const { return val;}

int intmod::get_modulo(){ return modulo;}

void intmod::set_modulo(int mod){ modulo = mod;}

intmod::operator +(int im) const{
    return intmod(val+im.val);
```

```
}
intmod &intmod::operator++(){
    val = (val-1)%modulo;
    return *this;
}

intmod intmod::operator++(int){
    intmod im=*this;
    val = (val+1)%modulo;
    return im;
}
```

Esercizio 2

Determinare cosa stampa il seguente programma:

Es2.cpp

```
class A{
private:
    int x;

public:
    A(int k=5): x(k)        // costruttore di default
    {
        cout << k << "A01";
    }
    A(const A&a): x(a.x)    { cout << "AC";}    //construttore di copia
    A g() const {return *this;}
};

class B{
private:
    A ar[2];    //array di due oggetti di tipo A
    static A a;

public:
    B(){ar[1] = ar(7); cout << "BA";}
};

A B::a = A(9);

int main()
{
    cout << "Zero" << endl;
    A a1; cout << "Uno" << endl;
    A a2(3); cour << "Due" << endl;
    A *b = &a1; cout << "Tre" << endl;
    B b; cout << "Quattro";
}
```

Output

```
9 A01 AC ZERO
5 A01 UNO
3 A01 DUE
TRE
5 A01 5 A01 7 A01 B0 QUATTRO
```

Esercizio 3

Il seguente programma non compila correttamente. Correggerlo:

Es3.cpp

```
class C{
public:
    int *const p;
    c(int A=0) : p(new int(A)) {}
};

int main()
{
    C x(3);
    C y;
    x = y;  /* ERRORE, non abbiamo ridefinito l'assegnazione, quindi viene
invocata l'assegnazione standard. Perciò eliminiamo questa riga. */
    C z(y);
}
```

Esercizio 4

Cosa stampa il seguente programma?

```
class C{
private:
    int d;

public:
    C(string s = ""): d(s.size()) {}
    explicit C(int m): d(m) {}
    operator int() {return d;}
    C operator+(C x) { return C(d+x.d);}
};

int main()
{
    C a, b("pippo"), c(3);
    cout << a << " " << 1+b << " " << c+4 << " " << c+b;
}
```

a: d = 0
b: d = 5
c: d = 3

OUTPUT

```
0 6 7 7
```

Esercizio 5

Cosa stampa il seguente programma?

Es5.cpp

```
class C{
public:
    int x;
    C(int k = 5): x(k) {}
    C *m (C&c) {
        if ((c.x != 5) && (x == 5))
            return &c;
        else
            return this;
    }
};

int main()
{
    C a, b(2), c(a);
    cout << (b.m(b))->x << " " <<
            (a.m(a))->x << " " << c.m(a) <<
            " " << c.m(c);
}
```

Output

```
2 5 (indirizzo di c) (indirizzo di c)
```