Laborator 4

Problemele se incarca pe Moodle pana duminica– 8.11.2020 ora 23:59.

Fisierele sursa ce contin rezolvarile o sa aiba numele:
Problema_x_Nume_Prenume_Grupa.cpp, unde x e numarul problemei, Nume, Prenume si Grupa – se refera la numele vostru si grupa din care faceti parte.

** Exercitiile se realizeaza astfel incat sa existe un fisier header si un fisier cpp pentru fiecare clasa si un fisier separat pentru main

```
class fractie
{
  private:
       int a; //numarator
       int b; //numitor


   public:

        fractie(int aa=0,int bb=0); // constructor cu parametrii impliciti
        fractie(const fractie&);// constructor de copiere
        fractie & operator=(const fractie &);
        //se intoarce referinta la obiectul modificat pt a putea face op de genul : int a,b,c,d ;
a=(b=(c=(d=4))) ;
        //asociativitate la dreapta
        ~fractie();
        // constr de copiere, op= si destr se genereaza automat si functioneaza corect
        // implementarea lor va fi facuta doar in scop didactic

         double getValoare();//cat face a/b
         fractie getInv();//b/a
         void setdata(int,int);//modifica valoarile numaratorului si numitorului
         float getA();//returneaza numaratorul
         float getB();//returneaza numitorul

         friend fractie operator +(const fractie &, const fractie&);  // supradefinire operator adunare
         friend fractie operator -( const fractie&, const fractie& );
         friend fractie operator *( const fractie&, const fractie&);
         friend fractie operator /( const fractie&, const fractie&);
         friend fractie operator -(const fractie&); //transforma numerele in inversul lor. ex: 8 -> -8
```

```
        fractie& operator +=( const complex& a){ // supradefinire operator incrementare cu o
valoare
        //se intoarce referinta la fractie pt a putea face operatii ca : int m ;  (m+=5)+=3 ;
                *this=*this+a;
        }
        fractie& operator -=( const fractie&);
        fractie& operator *=( const fractie&);
        fractie& operator /=( const fractie&);

        bool operator ==(const fractie &);//supradefinire operator de egalitate
        bool operator !=(const fractie& x);// supradefinire operator diferit
        // pot sa folosesc in implementare operatorul == implementat anterior {return(!(*this==x));}

        bool operator <(const fractie&);// supradefinire operator <
        bool operator >=(const fractie&);// supradefinire operator <
        bool operator >(const fractie&);// supradefinire operator >
        bool operator >=(const fractie&);// supradefinire operator <
};
```

Cerinta este sa implementati metodele si sa le testati in main.