```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace uppgift_3b
{
    class Program
    {
        static void Main(string[] args)
        {


            Fraction a = new Fraction(5, 5);
            Fraction b = new Fraction(10, 10);
            Fraction c = new Fraction(2, 7);
            Fraction d = new Fraction(1, 3);
            Fraction e = new Fraction(-2, -7);

            Console.WriteLine("Skriver ut ett bråk tal {0}",c);

            var cd = c + d;
            Console.WriteLine("addera ut ett bråk tal {0}", cd);

            var bd = b * d;
            Console.WriteLine("multiplicera ut ett bråk tal {0}\n", bd);

            Console.WriteLine("Tal i decimal form");
            Console.WriteLine((double)(a * b + c));
            Console.WriteLine();

            var ae = a + e;
            Console.WriteLine("Negativt bråk? {0}", bd.isNegative());
            Console.WriteLine("Negativt bråk? {0}\n", ae.isNegative());

            Console.WriteLine("Kolla om de är lika");
            Console.WriteLine(a.isEqualTo(b));
            Console.WriteLine(c.isEqualTo(b));

            Console.ReadLine();

        }
    }
}



namespace uppgift_3b
{
    class Fraction
    {
        int _numerator; // Täljare
        int _denominator; // Nämnare

        public Fraction(int num, int den)
        {
            getNumerator = num;
            getDenominator = den;
        }
        public int getNumerator
        {
            get{ return _numerator; }
            set{ _numerator = value; }

        }
        public int getDenominator
        {
            get{ return _denominator; }
            set{
                    if (value == 0)
                    {
                        throw new ArgumentException("Nämnaren får inte
vara 0");
                    }

                _denominator = value;
                }
        }

        //skapa Metoderna add och multiply // msdn.microsoft.com/en-
us/library/s53ehcz3.aspx
        public static Fraction operator +(Fraction first, Fraction
second)
        {
            int newNumerator = first.getNumerator * second.getDenominator
+ second.getNumerator * first.getDenominator;
            int newDenominator = first.getDenominator *
second.getDenominator;

            return new Fraction(newNumerator, newDenominator);
        }
        public static Fraction operator *(Fraction first, Fraction
second)
        {
            int newNumerator = first.getNumerator * second.getNumerator;
            int newDenominator = first.getDenominator *
second.getDenominator;

            return new Fraction(newNumerator, newDenominator);
        }
        //4.skapa metoden isNegative
        public bool isNegative()
        {

            //Undersök om bråktalet är negativt
            if (_numerator < 0 || _denominator < 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        //Kolla om bråk instanserna är de samma
        public bool isEqualTo(Fraction compare)
        {
            return (decimal)this.getNumerator / this.getDenominator ==
(decimal)compare.getNumerator / compare.getDenominator;
```

```
        }

        //Skriva ut ett bråk tal
        public override string ToString()
        {
            return string.Format("Tal: {0}/{1}", _numerator,
_denominator);
        }
        // Om bråket ska skrivas ut i decimal tal
        public static implicit operator double(Fraction f)
        {
            return (double)f.getNumerator / f.getDenominator;
        }
        //Räkna ut bråk tal
        public Fraction Inverse()
        {
            return new Fraction(getDenominator, getNumerator);
        }

    }
}
```