[1a]

```
namespace RaknaA
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Mata in en textrad: ");
            string input = Console.ReadLine();

            int countLower = 0;
            int countUpper = 0;

            foreach (char c in input)
            {
                if (c == 'a')
                {
                    countLower++;
                }
                if (c == 'A')
                {
                    countUpper++;
                }
            }
            Console.Write("Antal a: {0} Antal A: {1}\n", countLower, countUpper);
        }
    }
}
// Referenser:
// http://msdn.microsoft.com/en-us/library/x9h8tsay.aspx
// http://stackoverflow.com/questions/541954/how-would-you-count-occurrences-of-a-string-
within-a-string
// http://stackoverflow.com/questions/541954/how-would-you-count-occurrences-of-a-string-
within-a-string/541976#541976
```

[1b]

```
namespace RaknaSiffror
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Mata in ett heltal: ");
            string input = Console.ReadLine();

            int zero = 0;
            int odd = 0;
            int even = 0;

            // Alternativ 1
            //foreach (char c in input)
            //{
            //    if (c == '0') { zero++; }
            //    if (c == '1') { odd++; }
            //    if (c == '3') { odd++; }
            //    if (c == '5') { odd++; }
            //    if (c == '7') { odd++; }
            //    if (c == '9') { odd++; }
            //    if (c == '2') { even++; }
            //    if (c == '4') { even++; }
            //    if (c == '6') { even++; }
            //    if (c == '8') { even++; }
            //}

            // Alternativ 2
            foreach (var num in input)
            {
                if (num == '0')
                {
                    zero++;
                }
                else if (num % 2 == 1)
                {
                    odd++;
                }
                else
                {
                    even++;
                }
            }
            Console.Write("Nollor: {0} Udda: {1} Jämna: {2}\n", zero, odd, even);
        }
    }
}
// Referenser:
// http://stackoverflow.com/questions/160930/how-do-i-check-if-an-integer-is-even-or-odd
// http://cc.davelozinski.com/c-sharp/fastest-way-to-check-if-a-number-is-odd-or-even
```

[1c]

```
namespace NastStorsta
{
    class Program
    {
        static void Main(string[] args)
        {
            int amountOfTimes = 10;
            int largest = 0, sLargest = 0, count = 0;
            int input;

            Console.WriteLine("Mata in " + amountOfTimes + " heltal:\n");

            while (count < amountOfTimes)
            {
                try
                {
                    count++;
                    Console.Write(count+": ");
                    input = int.Parse(Console.ReadLine());

                    if (input > largest)
                    {
                        sLargest = largest;
                        largest = input;
                    }
                    else if (input < largest && input > sLargest)
                    {
                        sLargest = input;
                    }
                }
                catch
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.WriteLine("FEL! Ange heltal.");
                    Console.ResetColor();
                }
            }
            Console.WriteLine("\nDet näst största talet är: {0}\n",sLargest);
        }
    }
}
// Referenser:
// http://stackoverflow.com/questions/160930/how-do-i-check-if-an-integer-is-even-or-odd
```

[3a]

```
namespace Palindrom
{
    class Program
    {
        static void Main(string[] args)
        {
            while (true)
            {
                string input, reverse = "";

                Console.Write("Skriv en textrad: ");
                input = Console.ReadLine();

                if (input == "")
                {
                    Console.BackgroundColor = ConsoleColor.Red;
                    Console.WriteLine("Text raden är tom\n");
                    Console.ResetColor();
                }
                else if (input.Any(c => char.IsUpper(c)))
                {
                    Console.BackgroundColor = ConsoleColor.Red;
                    Console.WriteLine("Skriv in textraden enbart med små bokstäver.\n");
                    Console.ResetColor();
                }
                else
                {
                    for (int j = input.Length - 1; j >= 0; j--)
                    {
                        reverse += input[j].ToString();
                    }

                    if (reverse == input)
                    {
                        Console.ForegroundColor = ConsoleColor.Green;
                        Console.WriteLine("\"{0}\" är en palindrom.\n", input);
                        Console.ResetColor();
                        break;
                    }
                    else
                    {
                        Console.ForegroundColor = ConsoleColor.Red;
                        Console.WriteLine("\"{0}\" är inte en palindrom.\n", input);
                        Console.ResetColor();
                    }
                }
            }
        }
    }
}
// [Referenser]
// http://stackoverflow.com/questions/20032450/detect-if-a-string-contains-uppercase-characters
// http://www.c-sharpcorner.com/Blogs/13822/program-to-check-whether-a-string-palindrome-is-or-not.aspx
```

[3b]

```
namespace Fraction
{
    class Program
    {
        static void Main(string[] args)
        {
            // Test - Exempel på bråktal
            Fraction firstFraction = new Fraction(1, 3);
            Fraction secondFraction = new Fraction(-1, 3);

            // Addition test
            Console.WriteLine("Addition: {0}/{1} + {2}/{3}: \n" +
Fraction.add(firstFraction, secondFraction) + "\n",
                firstFraction.Numerator, firstFraction.Denominator,
secondFraction.Numerator, secondFraction.Denominator);

            // Multiplikation test
            Console.WriteLine("Multiplikation: {0}/{1} * {2}/{3}: \n" +
Fraction.multiply(firstFraction, secondFraction) + "\n",
                firstFraction.Numerator, firstFraction.Denominator,
secondFraction.Numerator, secondFraction.Denominator);

            // isEqualTo Test
            //Console.WriteLine("Representerar samma bråktal? \n{0}/{1} och {2}/{3}: " +
Fraction.isEqualTo(firstFraction, secondFraction) + "\n",
            //    firstFraction.Numerator, firstFraction.Denominator,
secondFraction.Numerator, secondFraction.Denominator);
        }
    }
}
```

```
namespace Fraction
{
    class Fraction
    {
        private int _denominator; // Nämnare
        private int _numerator; // Täljare

        // Konstruktor som skapar och initialiserar ett nytt bråktal.
        public Fraction(int numerator, int denominator)
        {
            Numerator = numerator;
            Denominator = denominator;
        }

        // Metoden getNumerator som returnerar täljaren.
        public int Numerator
        {
            get { return _numerator; }
            set { _numerator = value; }
        }
```

```
        // Metoden getDenominator som returnerar nämnaren. Nämnaren får inte vara noll.
        public int Denominator
        {
            get { return _denominator; }
            set
            {
                try
                {
                    if (value == 0)
                    {
                        _denominator = 1;
                        throw new ArgumentException();
                    }
                    _denominator = value;
                }
                catch (ArgumentException)
                {
                    Console.BackgroundColor = ConsoleColor.Red;
                    Console.WriteLine("{0}/0! Nämnaren får inte vara noll!", _numerator);
                    Console.ResetColor();
                }
            }
        }

        // Metoden isNegative som ger true om det är ett negativt bråktal.
        public bool isNegative()
        {
            if (_numerator < 0 || _denominator < 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        // Metoden add som adderar den första bråktalet med den andra och sedan returnerar
ett nytt bråktal.
        public static Fraction add(Fraction firstFraction, Fraction secondFraction)
        {
            Fraction result;

            // Gemensam nämnare
            if (firstFraction._denominator == secondFraction._denominator)
            {
                int newNumerator = firstFraction.Numerator + secondFraction.Numerator;

                // Ingen skillnad på nämnaren (samma som förut)
                result = new Fraction(newNumerator, firstFraction.Denominator);
            }
            // Ingen gemensam nämnare
            else
            {
                int newNumerator = (firstFraction.Numerator * secondFraction.Denominator)
+ (firstFraction.Denominator * secondFraction.Numerator);
                int newDenominator = firstFraction.Denominator *
secondFraction.Denominator;
```

```
                result = new Fraction(newNumerator, newDenominator);
            }
            return result;
        }

        // Metoden multiply som multiplicerar den först bråktalet med den andra bråktalet
och returnerar ett nytt bråktal.
        public static Fraction multiply(Fraction firstFraction, Fraction secondFraction)
        {
            int newNumerator = firstFraction.Numerator * secondFraction.Numerator;
            int newDenominator = firstFraction.Denominator * secondFraction.Denominator;

            return new Fraction(newNumerator, newDenominator);
        }

        // isEqualTo som jämför två Fraction-instanser och ser om de representerar samma
bråktal.
        public static bool isEqualTo(Fraction firstFraction, Fraction secondFraction)
        {
            if (firstFraction == secondFraction)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        // toString som returenerar en strängpresentation av bråktalet på form T/N.
        public override string ToString()
        {
            if (isNegative())
            {
                return String.Format("Resultat: {0}/{1} \nBråktalet är negativt",
_numerator, _denominator);
            }
            else
            {
                return String.Format("Resultat: {0}/{1} \nBråktalet är positivt",
_numerator, _denominator);
            }
        }
    }
}
```