Feladatok

1. Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti, hogy prímszám-e, vagy sem.

```
public static bool prim (int n)
 {
    if ( n <0)
    {
       Console.WriteLine("Nem természetes szám a megadott szám!");
       return false;
    } else
    {
       for (int i = 2; i <= Math.Sqrt(n); i++)
          if (n % 1 == 0) return false;
       return true;
    }

 }
```

2. Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti, hogy tökéletes szám-e, vagy sem! (pozitív osztóinak összege a szám kétszerese).

```
public static bool tokeletese(int n)
    {
       int osszeg = 1;

       if (n <0)
       {
          Console.WriteLine("A megadott szám nem a természetes számok közé tartozik!");
          return false;
       } else
       {
          for (int oszto =2; oszto <= n /2; oszto++)
          {
             if (n % oszto == 0) osszeg += oszto;

          }
          if (osszeg == n) return true; else return false;

       }

    }
```

3. Írjon olyan függvényt vagy metódust, amely egy karakterláncban vagy sztringben véletlenszerűen összekeveri a karaktereket (véletlenszám–generátor használható)!

```
public static string karakterkevero(string text)

    {

       Random rnd = new Random();

       StringBuilder sbIn = new StringBuilder(text);

       StringBuilder sbOut = new StringBuilder();

       for (int ix = text.Length; ix > 0; ix--)

       {
```

```
            int rNumber = rnd.Next(ix);
            sbOut.Append(sbIn[rNumber]);
            sbIn.Remove(rNumber, 1);
        }
        return sbOut.ToString();
    }
```

4. A következő közelítő formulát használva írjon függvényt vagy metódust, amely egy valós szám négyzetgyökét adja vissza! Használja az xk+1=1/2*(xk+a/xk) sorozatot, amely a négyzetgyökéhez konvergál, ha x1=1.

```
public static double negyzetgyoket(double szam)
    {
        double gy, ujgy, d, eps;
        gy = 1.0;
        d = 1.0;
        eps = 0.00001;
        while (d > eps)
        {
            ujgy = (1.0 / 2) * (gy + (szam / gy));
            d = ujgy - gy;
            if (d < 0) d = -d;
            gy = ujgy;
            return gy;
        }
        return 0;
    }
```

5. Írjon függvényt vagy metódust, amely egy valós szám köbgyökét adja vissza! Használja az xk+1=1/3*(2*xk+a/xk 2) sorozatot, amely a köbgyökéhez konvergál, ha x1=1.

```
public static double kobgyok(double szam)
    {
        int n = 20;
        double[] tomb = new double[n];
        tomb[0] = 1;
        for (int i = 1; i < n; i++)
        {
            double ertek = 0.3333333333333333 * (2 * tomb[i - 1] + szam / (tomb[i - 1] * tomb[i - 1]));
            tomb[i] = ertek;
        }

        double kimenet = tomb[n - 1];

        return kimenet;
    }
```

6. Írjon függvényt vagy metódust, amely kiszámolja az n-edik Fibonacci számot! A Fibonacci sorozatot az an=an-2+an-1 rekurzióval definiálja (n>2), ahol a1=a2=1.

```
public static int fibsor(int n)
   {
      int szam;
      int[] tomb = new int[n];
      tomb[0] = 1;
      tomb[1] = 1;
      for (int i = 2; i < n; i++)
      {
         tomb[i] = tomb[i - 2] + tomb[i - 1];
      }
      szam = tomb[n - 1];

      return szam;
   }
```

7. Írjon olyan függvényt vagy metódust, amely egy természetes számhoz visszaadja azt a legnagyobb egész kitevős hatványát, amely még éppen kisebb, mint 567!

```
 public static int hatvany(int n)
   {
      int szam = n;
      while (szam * n < 576)
      {
         szam = szam * n;
      }

      return szam;
   }
```

8. Írjon olyan függvényt vagy metódust, amely egy természetes szám esetén kiírja, hogy a 9-es számjegyből hány darabot tartalmaz (ne alakítsa át sztringgé/karaktertömbbé)!

```
public static int hanyKilences(int szam)
   {

      int count = 0;
      int maradek = 0;

      while (szam > 0)
      {

         maradek = szam % 10;
         if (maradek == 9) count++;
         szam = szam / 10;

      }

      return count;

   }
```

9. Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti, hogy a kettes számrendszerbeli felírásában a jobbról második bitje 1 vagy 0 (ne alakítsa át sztringgé/karaktertömbbé)!

```
public static int jobbrolMasodikBit(int szam)
  {

     int szam2 = szam / 2;
     return szam2 % 2;

  }
```

10. Írjon olyan függvényt vagy metódust, amelynek paramétere egy 1 < x < 10 természetes szám, és kiírja az 1,3,4,6,7,9,10,12,... sorozat első öt x-szel osztható elemét, azaz a sorozat i+1-edik tagja 2-vel nagyobb az i-ediknél, ha i páratlan, s eggyel nagyobb az i-ediknél, ha i páros!

```
public static void elso5(int szam)
  {

     if (szam > 1 && szam < 10)
     {

        int y = 1;
        int yUtan = 1;
        int count = 0;
        int index = 1;

        while (count < 5)
        {

           y = yUtan;

           if (index % 2 == 1)
           {
              yUtan = y + 2;
           }
           else
           {
              yUtan = y + 1;
           }

           if (yUtan % szam == 0)
           {
              Console.WriteLine(yUtan + ", ");
              count++;
           }
           index++;
        }
     }
     else
     {
        Console.WriteLine(" A megadott szám nem 1 és 10 között van!");
     }

  }
```

11. Írjon olyan függvényt vagy metódust, amely a paraméterében megadott természetes szám pozitív osztóinak számával tér vissza!

```
public static int pozitivOsztokSzama(int szam)
  {

    int count = 0;

    for (int i = 1; i <= szam; i++)
    {

      if (szam % i == 0)
      {

        count++;
        //Console.WriteLine(i);

      }

    }

    return count;

  }
```

12. Írjon olyan függvényt vagy metódust, amely egy karakterláncból vagy sztringből a számjegyek kivételével minden karaktert eltávolít!

```
public static String onlyNumbers(String s)
  {

    StringBuilder newString = new StringBuilder();

    for (int i = 0; i < s.Length; i++)
    {

      if (s[i] > 47 && s[i] < 58)
      {

        newString.Append(s[i]);

      }

    }

    return newString.ToString();

  }
```

13. Írjon olyan függvényt vagy metódust, amely egy karakterláncról vagy sztringről eldönti, hogy palindróma-e! (Balról olvasva ugyanaz, mint jobbról olvasva.)

```
public static bool palindromaE(String s)
    {

        StringBuilder szo = new StringBuilder();
        StringBuilder szoForditva = new StringBuilder();

        for (int i = 0; i < s.Length; i++)
        {

            if (s[i] != ' ')
            {

                szo.Append(s[i]);

            }

        }

        for (int j = szo.Length; j > 0; j--)
        {

            szoForditva.Append(szo[j - 1]);

        }

        return szo.ToString().Equals(szoForditva.ToString());

    }
```

14. Írjon olyan függvényt vagy metódust, amely egy, az angol ábécé betűit tartalmazó karakterláncban vagy sztringben minden szó kezdőbetűjét nagybetűre alakítja!

```
static String wordsToUppercase(String s)
    {

        String[] words = s.Split("\\s");
        StringBuilder newString = new StringBuilder();

        for (int i = 0; i < words.Length; i++)
        {

            newString.Append(words[i].Substring(0, 1).ToUpper() + words[i].Substring(1) + " ");

        }

        return newString.ToString();

    }
```

15. Írjon olyan függvényt vagy metódust, amely egy karakterláncból vagy sztringből eltávolítja egy megadott karakter összes előfordulását!

```
static String deleteThisChar(String s, char c)
    {

        string uj = "";
        foreach (char aktual in s)
        {
            if (aktual != c)
            {
                uj += aktual;
            }
        }
        return uj;

    }
```

16. Írjon olyan függvényt vagy metódust, amely megszámolja egy adott karakterlánc vagy sztring összes előfordulását egy másik karakterláncban, vagy sztringben

```
public static int incidence(String String, String subString)
    {

        int counter = 0;
        string newString = String.ToLower();
        int i;

        while (newString.IndexOf(subString) != -1)
        {

            i = newString.IndexOf(subString);
            newString = newString.Substring(i + subString.Length);
            // System.out.println( newString );

            counter++;

        }

        return counter;

    }
```

17. Írjon olyan függvényt vagy metódust, amely kiírja az angol kisbetűs ábécé azon betűit, melyek ASCII kódja négyzetszám!

```
public static void squareASCII()
    {

      // 65 A -> 122 z
      int a = (int)'a';
      int z = (int)'z';

      for (int i = a; i <= z; i++)
      {


        if (Math.Pow((int)Math.Sqrt(i), 2) == i)
        {

          Console.WriteLine((char)i + " ");

        }

      }

      Console.WriteLine("\n");

    }
```

18. Írjon olyan függvényt vagy metódust, amely előállít egy 5 karakterből (angol kisbetűs ábécé karaktereit használva) álló véletlen karakterláncot vagy sztringet! Biztosítsa, hogy minden 5 hosszú különböző betűkből álló sztring egyenlő valószínűséggel kerüljön kiválasztásra, feltéve, hogy a választott programozási nyelv véletlenszám-generátora egyenletes eloszlást biztosít!

```
public static String randomString()
    {

      int a = (int)'a';
      int z = (int)'z';
      int rndInt;
      Random rand = new Random();
      StringBuilder sb = new StringBuilder();

      Console.WriteLine(a + ", " + z);

      for (int i = 0; i < 5; i++)
      {

        rndInt = Math.Abs((int)rand.Next() % (z - a + 1)) + a;
        sb.Append((char)rndInt);

      }

      return sb.ToString();

    }
```

19.Írjon olyan függvényt vagy metódust, amely egy karakterláncba vagy sztringbe beszúr egy „a” karaktert véletlenül választott pozícióba (véletlenszám–generátor használható)!

```
public static void beszur(String s)
  {
  StringBuilder sb = new StringBuilder();
     Random r = new Random();
     int a = r.Next(s.Length);
     for (int i = 0; i < s.Length; i++)
     {
        if (i != a)
        {
           sb.Append(s[i]);
        }
        else
        {
           sb.Append('a');
           sb.Append(s[i]);
        }
     }
      Console.WriteLine(sb.ToString());

  }
```

20.Adjon olyan függvényt vagy metódust, ami adott két pozitív egész paramétere esetén megadja (n alatt a k)=n!/k!(n-k)! értékét. Használjon rekurziót!

```
public static string kombinacio(int n, int k)
   {
      int nfakt = 1, kfakt = 1, nkfakt = 1;
      int eredmeny = 1;
      if (n < 0 || k < 0)
      {
         return "A két megadott szám közül legalább az egyik negatív szám!";
      }
      else
      {
         for (int i = 1; i < n + 1; i++)
         {
            nfakt = nfakt * i;
         }
         for (int x = 1; x < k + 1; x++)
         {
            kfakt = kfakt * x;
         }
         int nk = n - k;
         for (int w = 1; w < nk + 1; w++)
         {
            nkfakt = nkfakt * w;
         }
         eredmeny = nfakt / (kfakt * nkfakt);
         return eredmeny.ToString();
      }

   }
```

21.Adjon olyan metódust vagy függvényt, ami eldönti, hogy a paramétereként megadott (pozitív egészekből álló) nemüres tömbben van-e olyan szám, ami az összes többit osztja. (Maradékszámító függvény használható).

```
public static bool vanEMindendOszto(int[] arr)
  {
     bool b = false;
     int counter = 0;
     for (int i = 0; i < arr.Length; i++)
     {
        counter = 0;
        for (int j = 0; j < arr.Length; j++)
        {
           /* hogyha a vizsgált szám ossza a tömmben helyet foglaló j-edik számot,
           akkor növeljük a számlálót */
           if (arr[j] % arr[i] == 0) counter++;
        }
        /* ha a számláló megegyzik a több méretével, az azt jelenti,
        hogy az összes számot ossza a tömbben, ez esetben a bool változot true-ra módosítjuk */
        if (counter == arr.Length) b = true;
        /* a számlálót visszaállítjuk 0-ra a következő ciklushoz */
     }
     /* ha a bool átvált valamikor is true-ra, akkor van egy olyan szám,
     amelyik mindent oszt a tömbben */
     return b;
  }
```

22.Adjon olyan metódust vagy függvényt, ami eldönti, hogy a paramétereként megadott (pozitív egészekből álló) nemüres tömbben van-e olyan szám, ami az összes többinél többször fordul elő.

```
static bool vanELeggyakrabbSzam(int[] arr)
 {
    int maxElofordulas = 0;
    int elofordulas = 0;
    int legtobbszorEloforduloSzam = 0;
    bool b = false;
    for (int i = 0; i < arr.Length; i++)
    {
       for (int j = 0; j < arr.Length; j++)
       {
          if (arr[i] == arr[j])
             elofordulas++;
       }
       if (elofordulas > maxElofordulas && i != 0 && arr[i] != legtobbszorEloforduloSzam)
       {
          maxElofordulas = elofordulas;
          legtobbszorEloforduloSzam = arr[i];
          b = true;
       }
       else if (elofordulas == maxElofordulas && arr[i] != legtobbszorEloforduloSzam)
       {
          b = false;
       }
       elofordulas = 0;}
    return b;}
```

23.Adjon olyan metódust vagy függvényt, ami visszaadja, hogy a paramétereként megadott (pozitív egészekből álló) nemüres tömbben melyik index az, ahol a leghosszabb folyamatosan növekvő részsorozat kezdődik. Ha több ilyen index is van, az utolsót adja vissza.

```
public static int reszSorozatIndex(int[] arr)
{
   int kezdoIndex = 0,
      sorozatHossz = 0,
      leghosszabbSorozatHossz = 0;
   for (int i = 0; i < arr.Length - 1; i++)
   {
      /* ha az i-edik szám egyenlő a következő szám-1 */
      if (arr[i] == arr[i + 1] - 1)
      {
         /* az aktuális sorozat hossza így egyel növekszik */
         sorozatHossz++;
         /* ha ez a sorozat így eddig a leghosszabb */
         if (sorozatHossz > leghosszabbSorozatHossz)
         {
            /* akkor ez a leghosszabb */
            leghosszabbSorozatHossz = sorozatHossz;
         }
         /* ha az utolsó előtti elemet nézzük és az így kialakult sorhossz egyenlő,
         vagy nagyobb ... */
         if (i == arr.Length - 2 && sorozatHossz >= leghosszabbSorozatHossz)
         {
            kezdoIndex = i - sorozatHossz + 1;
         }
      }
      else
      {
         if (sorozatHossz >= leghosszabbSorozatHossz)
         {
            kezdoIndex = i - sorozatHossz;
         }
         sorozatHossz = 0;
      }
   }
   return kezdoIndex;
}
```

24.Adjon olyan metódust vagy függvényt, ami visszaadja, hogy a paramétereként megadott (pozitív egészekből álló) nemüres tömbben melyik az a legkisebb index, amire az index előtti elemek összege meghaladja a tömb első két elemének szorzatát. Ha nincs ilyen, 0-t adjon vissza.

```
public static int whichIndex(int[] arr)
   {
      int smallestIndex = 0;
      int multipOfFirstTwoElement = arr[0] * arr[1];
      int sum = 0;
      for (int i = 2; i < arr.Length; i++)
      {
         sum = 0;
         for (int j = 0; j < i - 1; j++)
         {
            sum += arr[j];
```

```
        }
        if (sum > multipOfFirstTwoElement)
        {
            smallestIndex = i;
            return smallestIndex;
        }
    }
    return smallestIndex;
}
```

25.Adjon egy metódust vagy függvényt, ami paraméterként adott s sztring/karaktertömb, c karakter és n pozitív egész szám esetén megadja, hogy a c karakter n-edik előfordulása hányadik pozíción van az „s” sztringben.

```
public static int elofordulas(String s, char c, int n)
{

    int counter = 0, poz = 0;

    for (int i = 0; i < s.Length; i++)
    {
        if (s[i] == c)
        {
            counter++;
            if (counter == n)
            {
                poz = i;
            }
        }
    }
    return poz;
}
```

26.Adjon metódust vagy függvényt, ami a paraméterként kapott, egészekből álló rendezett tömbben a tömb hosszának logaritmusával arányos lépésszám alatt megkeresi a paraméterként kapott egész első előfordulásának indexét, illetve ha nincs ilyen, akkor -1-et ad vissza. (pl. a bináris keresés)

```
public static int elofordulas(int[] szamok, int keresett)
    {
       int bal = 0, jobb = szamok.Length - 1, kozep;
       do
       {
          kozep = (bal + jobb) / 2;
          if (keresett > szamok[kozep])
          {
             bal = kozep + 1;
          }
          if (keresett < szamok[kozep])
          {
             jobb = kozep - 1;
          }
       } while (bal <= jobb && keresett != szamok[kozep]);

       if (keresett == szamok[kozep])
       {
          return kozep;
       }
       else
       {
          return -1;
              }
        }
```

27.Írjon függvényt vagy metódust, mely visszaadja két egész paramétere szorzatának balról második számjegyét! (a megoldás során ne használjon sztringeket/karaktertömböket)

```
public static int masodikSzam(int num1, int num2)
    {

       int szam = num1 * num2;

       while (szam > 99)
       {

          szam /= 10;

       }

       return szam % 10;

    }
```

28.Írjon függvényt vagy metódust, mely eldönti, hogy a paraméterként kapott 5x5-ös /karakterekből álló/ tömbben a főátlóban van-e olyan elem, mely a főátlón kívül is megjelenik a tömbben!

```
public static bool megjelenike(char[][] tomb)
    {
       for (int i = 0; i <= 4; i++)
       {
          for (int j = 0; j <= 4; j++)
          {
             for (int k = 0; k <= 4; k++)
             {
                if (i != j && tomb[i][j] == tomb[k][k])
                {
                   return true;
                }
             }
          }
       }
       return false;
    }
```

29.Írjon függvényt vagy metódust, mely valós típusú paraméterének azt a számjegyét adja vissza, amelyik a tizedes pont után áll! (a megoldás során ne használjon sztringeket/karaktertömböket)

```
public static int tizedesPontUtaniSzamjegy(double num)
    {

       int szam = (int)(num * 10);

       return szam % 10;

    }
```

30.Írjon függvényt vagy metódust, mely pozitív egész paraméterét fordítva adja vissza, pl. fordit(234) eredménye 432! (a megoldás során ne használjon sztringeket/karaktertömböket)

```
public static int fordit(int num)
    {

       int szam = num;
       int forditva = 0;

       while (szam > 0)
       {

          forditva = (forditva * 10) + (szam % 10);
          szam /= 10;

       }

       return forditva;

    }
```

31.Írjon függvényt vagy metódust, mely a paraméterként kapott 10x10-es mátrixról eldönti, hogy van-e olyan eleme, mely sorában nagyobb és oszlopában pedig kisebb a többi elemnél!

```
public static bool vanELegkisebbElem(int[][] tomb)
    {
       bool sorbanALegnagyobb = true;
       bool oszlopbanALegkisebb = true;
       for (int ri = 0; ri < 10; ri++)
       {
          for (int ci = 0; ci < 10; ci++)
          {
             sorbanALegnagyobb = true;
             oszlopbanALegkisebb = true;
             /* sorában a legnagyobb? */
             for (int i = 0; i < 10; i++)
             {
                if (tomb[ri][ci] < tomb[ri][i])
                {
                   sorbanALegnagyobb = false;
                }
             }
             /* oszlopában a legkisebb? */
             for (int j = 0; j < 10; j++)
             {
                if (tomb[ri][ci] > tomb[j][ci])
                {
                   oszlopbanALegkisebb = false;
                }
             }
             if (oszlopbanALegkisebb == true && sorbanALegnagyobb == true)
             {
                return true;
             }
          }
       }
       return false;
    }
```

32.Írjon függvényt vagy metódust, mely visszaadja, hogy k-tól m-ig hány olyan szám van, melyeknek n db valódi osztója van! (n, k és m paraméter).

```
public static int valodie(int k, int m, int n)
  {
     int szamlalo = 0;
     for (int i = k; i <= m; i++)
     {
        int osztokszama = 0;
        for (int j = 2; j < i; j++)
        {
           if (i % j == 0)
              osztokszama++;
        }
        if (osztokszama == n)
           szamlalo++;
     }
     return szamlalo;
  }
```

33.Írjon függvényt vagy metódust, mely visszaadja, hogy két pozitív egész paraméterének legkisebb közös többszöröse hány számjegyből áll kettes számrendszerben

```
public static int hanySzamjegyKettesben(int num1, int num2)
    {

        int lkkt = 1;
        int szamjegy = 0;
        bool b = true;
        while (b)
        {
            if (lkkt % num1 == 0 && lkkt % num2 == 0 && lkkt != num1 && lkkt != num2)
            {
                b = false;
            }
            else
            {
                lkkt++;
            }
        }
        while (lkkt > 0)
        {
            lkkt /= 2;
            szamjegy++;
        }
        return szamjegy;
    }
```