1. feladat:

Alkalmazás bezárása:

```
Application.Exit();
```

2. feladat:

Páros számok generálása:

```
label1.Text = "";
for (int i=1; i<=10; i++)
{
      label1.Text += (2 * i) + ", ";
}
```

Fibonacci számok: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

```
int a = 0, b = 1, c;
label1.Text = "0, 1, ";
for (int i = 3; i <= 10; i++)
{
      c = a + b;
      label1.Text += c + ", ";
      a = b;
      b = c;
}
```

3. feladat:

TextBox-ba beírt szöveg tárolása string típusú változóban:

```
string s = textBox1.Text;
```

4. feladat:

Véletlen számok generáláshoz előbb létrehozunk egy generátor objektumot:

```
Random randgen = new Random();
```

Majd a létrehozott generátor objektum segítségével így tudunk 1-90 közötti egész számot kigenerálni:

```
int r = randgen.Next(1, 91);
```

Ahhoz, hogy a kigenerált számok között ne legyen két egyforma, használjunk egy öt elemű tömböt:

```
int[] a = new int[5];
```

Minden egyes lottószám kigenerálása után nézzük meg, hogy található-e már ilyen szám (“r”) a tömbben (“a”).

- Ha nincs még ilyen szám a tömbben, tegyük bele, majd továbbléphetünk a következő lottószám generálására.
- Ha viszont már van ilyen szám a tömbben, generáljunk “r” helyett másikat mindaddig, amíg nem sikerül olyant kigenerálni, ami még nincs a tömbben.

A lottószámok kigenerálást tartalmazó programrész váza:

```
// öt lottószám generálása
for (int i=0; i<5; i++)
{
	int r;
	bool van;
	do
	{
		van = false;

		// kigeneráljuk “r” értékét

		// for ciklussal végignézzük az eddigi tömbelemeket
		// és megvizsgáljuk van-e “r” az “a” tömbben;
		// ha találunk ilyen elemet, akkor a “van” értékét
		// “true”-ra állítjuk

	} while (van);
	a[i] = r;
}
```

Miután sikerült kigenerálni mind az öt lottószámot, rendezzük a tömböt növekvő sorrendbe:

Array.Sort(a);

Rendezés után már csak írjuk ki a tömbelemeket egymás után a “label1”-be egy for ciklus segítségével.

```
label1.Text = “Számok: “;
for (int i=0; i<5; i++) {
	label1.Text += a[i] + “, “;
}
```

5. feladat:

Karakterlánc (string) átalakítása valós számmá:
```
double kg1 = Convert.ToDouble(textBox1.Text);
```

Az “osszes” változó értékének kikerekítése két tizedesjegyre:
```
osszes = Math.Round(osszes, 2);
```

6. feladat:

Vegyünk fel egy globális változót, amely azt fogja megadni, hogy éppen melyik sarokban van a gomb. Ennek kezdeti értéke legyen 0:
```
private int szamlalo = 0;
```

A gomb lenyomásakor növeljük meg ennek a számlálónak az értékét eggyel, majd attól függően, hogy ez 1, 2, 3 vagy 4, állítsuk be a gomb helyét az ablakban. Ha a számláló értéke 4, ne felejtsük el kinullázni.

Gomb helyének beállítása pl. az ablak 100, 50 koordinátáira:
```
button1.Location = new Point(100,50);
```

Az ablak bal felső sarkának koordinátái mindig `0, 0`.

Az ablak jobb alsó koordinátái a Form1 osztályon belül a `this.ClientSize.Width` és **this.ClientSize.Height** tulajdonságokkal állapítható meg.

A gomb szélességét és magasságát a **button1.Size.Width** és **button1.Size.Height** tulajdonságok adják meg.

7. feladat:

Vegyünk fel egy globális Random objektumot (amivel majd véletlen számokat tudunk generálni) és egy globális változót (amiben a gondolt számot fogjuk tároljuk):
```
private Random randgen = new Random();
private int gondoltSzam;
```

Az ablakunk (Form1) `Load` eseményéhez tartozó függvényben adjunk a `gondoltSzam` változónak 1-50 véletlen értéket! A `Load` esemény akkor következik be, amikor az ablakunk betöltődik (tehát amikor az alkalmazás indul).

Ez után már csak a gomb lenyomásához (`Click` esemény) tartozó függvényt kell megírni, ahol a felhasználó által beírt számot összehasonlítjuk a `gondolSzam` változó értékével.

8. feladat:

Használjuk a jelölőnégyzetek **CheckedChanged** eseményét! A **CheckedChanged** esemény akkor következik be, amikor a jelölőnégyzetet be- vagy kijelöli a felhasználó.

Azt, hogy egy jelölőnégyzet ki van-e jelölve, annak **Checked** tulajdonsága adja meg (értéke igaz vagy hamis). Mindegyik jelölőnégyzet **Checked** tulajdonságát le kell ellenőriznünk ahhoz, hogy tudjuk melyek vannak bepipálva.

Elegendő az első jelölőnégyzet **CheckedChanged** eseményéhez megírni a számítást végző függvényt (alapár + a kijelölt összetevők ára), majd a többi jelölőnégyzet **CheckedChanged** eseményéhez ugyanezt a függvényt beállítani.

9. feladat:

Az italok és a hozzávalók egy-egy **GroupBox** komponensben vannak (ez a komponens a Toolbox-ban a Containers kategória alatt található). Ezekbe a **GroupBox**-okba raktunk három **RadioButton** ill. négy **CheckBox** komponenst.

Azt, hogy egy komponens ne legyen elérhető (szürke legyen), annak **Enabled** tulajdonságával állíthatjuk be.

Amikor a felhasználó kiválaszt egy **RadioButton**-t, annak **Click** eseménye következik be.

Amikor a felhasználó kiválaszt egy **CheckBox**-ot, annak szintén **Click** eseménye következik be.

Azt, hogy egy **RadioButton** ki van-e jelölve, annak **Checked** tulajdonsága adja meg (értéke lehet igaz vagy hamis).

Azt, hogy egy **CheckBox** ki van-e jelölve, annak szintén a **Checked** tulajdonsága adja meg (értéke lehet igaz vagy hamis).

10. feladat:

A csúszkák a Toolbox-ban az All Windows Forms kategória alatt található **HScrollBar** komponensek.

A **HScrollBar** komponensek **Minimum** és **Maximum** tulajdonságával állítható be annak alsó és felső értéke. A csúszka felhasználó által beállított értékét a **Value** tulajdonság adja meg.

Amikor a felhasználó módosítja a **HScrollBar** értékét, annak **ValueChanged** eseménye következik be (ehhez írhatjuk a szükséges programkódot).

Ahhoz, hogy egy **Label** méretét és háttérszínét be tudjuk állítani úgy, ahogy a feladatban van, a következőket kell tennük:

- Kikapcsolni az automatikus méretezést (benne levő szöveghez igazodást) az **AutoSize** tulajdonsággal. Ez után a Label tetszőlegesen átméretezhető.
- A **Text** tulajdonságát beállítani üresre.
- Amikor módosítani szeretnénk a **Label** háttérszínét, annak a **BackColor** tulajdonságot fogjuk beállítani (kezdetben ezt állítsuk feketére).

Egy színt a **Color** osztály **FromArgb(r,g,b)** metódusával tudunk kikeverni, ahol r, g, b a piros, zöld, kék összetevő (0–255 értékeik lehetnek), pl.:
label1.BackColor = Color.FromArgb(255, 200, 50);

A mi esetünkben a három összetevő azonban nem a 255, 200, 50 számok lesznek, hanem a három csúszkán beállított értékek.

11. feladat:

A feladat hasonló az előzőhöz. Ügyeljünk arra, hogy 0-van nem lehet osztani (ekkor vagy egy hibaüzenetet jelenítsünk meg):

```
if (hScrollBar2.Value==0) {
  label3.Text = "Nullaval nem lehet osztani!";
} else {
  label3.Text = "Eredmény: "
      + ((double)hScrollBar1.Value / hScrollBar2.Value);
}
```

12. feladat:

A verem egy **ListBox** komponens segítségével legyen szemléltetve.

Használjuk a **ListBox** alábbi adatmezőit és metódusait:

- **listBox1.Items.Count** – a listában levő elemek számát adja meg
- **listBox1.Items.Insert(0, “egér”)** – az “egér” szöveg beszúrása a lista 0. helyére (a lista elemei 0-tól vannak sorszámozva)
- **listBox1.Items[0]** – a listában található 0. elemet kapjuk meg, melyet a mi programunkban valószínűleg még szöveggé kell majd alakítanunk így: **listBox1.Items[0].ToString()**
- **listBox1.Items.RemoveAt(0)** – a listából kitörli a 0. elemet