PHP függvények

Hozzon létre egy `fuggvenyek.php` és egy `test.php` fájlt. A függvényeket előbbibe, míg a tesztelést utóbbi fájlban végezze. A tesztelés során legalább az itt megadott példakódokkal mind működnie kell! Az első két feladatcsoportot ezeknek a fájloknak a felhasználásával oldja meg.

Egyszerű függvények

1. Készítsen egy függvényt `hetNapja` néven, ami a paraméterül kapott nap sorszámából megállapítja, hogy az a hét melyik napja. A kimenete szöveges típusú lesz. Például:

```php
echo hetNapja(1); // hétfő
echo hetNapja(5); // péntek
```

2. Készítsen egy függvényt `napSorszama` néven, ami a paraméterül kapott nap nevéből (string)megállapítja, hogy az a hét hányadik napja. A kimenete szám típusú lesz. Például:

```php
echo napSorszama("hétfő"); // 1
echo napSorszama("péntek"); // 5
```

3. Készítsen egy függvényt `parosE` néven, aminek egy bemeneti paramétere van, aminek a típusa szám. A függvény kimenete logikai, ami megállapítja, hogy a paraméterül kapott szám páros -e vagy sem.

```php
var_dump(parosE(5));  // bool(false)
var_dump(parosE(8));  // bool(true)
```

4. Készítsen egy függvényt `paratlanE` néven, aminek egy bemeneti paramétere van, aminek a típusa szám. A függvény kimenete logikai, ami megállapítja, hogy a paraméterül kapott szám páratlan -e vagy sem.

```php
var_dump(paratlanE(5));  // bool(true)
var_dump(paratlanE(8));  // bool(false)
```

5. Készítsen egy függvényt `oszthatoE` néven, aminek két egész szám a bemenete. A kimenete egy logikai érték, ami azt határozza meg, hogy az első paraméter osztható -e a második paraméterrel.

```php
var_dump(oszthatoE(5, 5));  // bool(true)
var_dump(oszthatoE(8, 5));  // bool(false)
```

6. Készítsen egy függvényt `negativE` néven, aminek egy bemeneti paramétere van, egy valós szám. A kimenete egy logikai érték, ami akkor igaz, ha a szám negatív, különben hamis.

```php
var_dump(negativE(-3));  // bool(true)
var_dump(negativE(96));  // bool(false)
```

7. Készítsen egy függvényt `szignum` néven. A bemenete egy valós szám, a kimenete a paraméterül kapott szám szignum értéke. Egy szám szignuma -1, ha a szám negatív, 0 esetén 0 lesz, egyéb esetben 1.

```
echo szignum(-836);     // -1
echo szignum(0);        // 0
echo szignum(1024);     // 1
```

8. Készítsen egy függvényt `datumIdo` néven, aminek egy szöveges paramétere van, méghozzá az, hogy a pontos idő melyik részére kíváncsi a felhasználó.

   Lehetséges bemeneti értékek: `"év"`, `"hónap"`, `"nap"`, `"óra"`, `"perc"`, `"másodperc"`.

   A kimenete a pontos időnek a kért része. Minden esetben a bevezető nullák is kerüljenek megjelenítésre (lásd példa)!

   A bevezető nullák miatt a függvény visszatérési értékének a típusa legyen szöveg.

   - https://www.php.net/manual/en/function.date.php

```
echo datumIdo("óra");       // 08
echo datumIdo("perc");      // 11
echo datumIdo("másodperc"); // 08
echo datumIdo("év");        // 2022
echo datumIdo("hónap");     // 09
echo datumIdo("nap");       // 06
```

Függvények tömbökön

Az alábbi feladatokban számokat tartalmazó, 0-tól indexelt tömbökön kell megvalósítani a függvényeket.

1. Készítsen egy függvényt `utolso` néven, ami paraméterül kap egy tömböt, és annak az utolsó elemével tér vissza. Tetszőleges típussal működjön! Például:

```
echo utolso([5,11,76,3]);   // 3
```

2. Készítsen egy függvényt `osszeg` néven, ami paraméterül kap egy tömböt, és ez alapján meghatározza az abban található számok összegét. Feltételezheti, hogy a tömb csak számokat tartalmazhat. Például:

```
echo osszeg([5,11,76,3]);   // 95
```

3. Készítsen egy függvényt `szorzat` néven, ami paraméterül kap egy tömböt, és ez alapján meghatározza az abban található számok szorzatát. Feltételezheti, hogy a tömb csak számokat tartalmazhat. Például:

```
echo szorzat([5,11,76,3]);   // 12540
```

4. Készítsen egy függvényt `parosDb` néven, ami paraméterül kap egy tömböt, és ez alapján meghatározza hány páros szám található benne. A feladat elvégzéséhez használja fel a korábban megírt `parosE` függvényt. Feltételezheti, hogy a tömb csak számokat tartalmazhat. Például:

```
echo parosDb([]);               // 0
echo parosDb([5,11,76,3]);      // 1
echo parosDb([37,74,3,71,54]);  // 2
```

5. Készítsen egy függvényt `parosOsszeg` néven, ami paraméterül kap egy tömböt, és ez alapján meghatározza a benne található páros számok összegét. A feladat elvégzéséhez használja fel a korábban megírt `parosE` függvényt. Feltételezheti, hogy a tömb csak számokat tartalmazhat. Például:

```
echo parosOsszeg([]);                // 0
echo parosOsszeg([5,11,76,3]);       // 76
echo parosOsszeg([37,74,3,71,54]);   // 128
```

6. Készítsen egy függvényt `elsoNOsszeg` néven, ami paraméterül kap egy tömböt, és egy `n` számot, ami 0 és a tömb utolsó indexe közötti szám. Ez alapján meghatározza a benne található számok összegét, de csak a megadott `n` indexig (az indexet is beleértve).Feltételezheti, hogy a tömb csak számokat tartalmazhat. Például:

```
echo elsoNOsszeg([5,11,76,3],2);       // 16
echo elsoNOsszeg([37,74,3,71,54],3);   // 114
```

Fokváltó

1. Hozzon létre egy `f2c` nevű függvényt, ami a kapott fahrenheit fokot celsius fokká váltja át.

   $C = \frac{F-32}{1,8}$

```
echo f2c(68);   // 20
```

2. Hozzon létre egy `c2f` nevű függvényt, ami a kapott celsius fokot fahrenheit fokká váltja át.

   $F = C \cdot 1,8 + 32$

```
echo c2f(30);   // 86
```

3. Hozzon létre egy fájlt `fokvalto.php` néven. A fájlnak két bemeneti paramétere van, az átváltandó fok és a a mértékegység. A mértékegység függvényében alkalmazza a `fuggvenyek.php` fájlban található `c2f` és `f2c` függvényeket a sikeres átváltáshoz. Az eredmény két tizedesre kerekítse! Elfogadott mértékegységek: `"c"`, `"C"`, `"celsius"`, `Celsius`, `CELSIUS`, `"f"`, `"F"`, `"fahrenheit"`, `"Fahrenheit"`, `"FAHRENHEIT"`

- http://php.net/manual/en/function.round.php

```
php fokvalto.php 25 Celsius
25 celsius = 77.00 fahrenheit
```

```
php fokvalto.php 25 f
25 fahrenheit = -3.89 celsius
```

4. Az elkészített feladatot dockerizálja. Az image neve legyen `monogram/fokvalto`.

```
$> docker run rcs/fokvalto 25 Celsius
25 celsius = 77.00 fahrenheit
```

Fogyasztás

1. Hozzon létre egy függvényt `fogyasztas` néven. A függvénynek két paramétere van:

   - `km` a legutóbbi tankolás óta megtett út kilométerben (egész)
   - `liter` a jelenlegi tankolásnál hány litert tanoltak (valós)

   A függvény határozza meg az átlagfogyasztás, az eredményt két tizedesre kerekítve adja vissza. Az átlagfogyasztás azt határozza meg, hogy 100 km megtételéház hány liter üzemanyagot használ fel az autó.

   ```
   $km = 620;
   $liter = 42.3

   echo fogyasztas($km, $liter);   // 6.82
   ```

2. Hozzon létre egy fájlt `fogyasztas.php` néven. A fájlnak három bemenete van.

   - Mennyi volt a kilométer óra állás a legutóbi tankoláskor
   - Mennyi a kilométer óra állása jelenleg
   - Most hány litert sikerült tankolni a járműbe

   Az első két adatból kiszámítható, hogy mennyit tett meg az autó, enek a felhasználásával és a tankolt mennyiségből megállapítható az átlagfogyasztást. Az eredményeket a minta szerint jelenítse meg!

   - https://www.php.net/manual/en/function.number-format.php

   ```
   php fogyasztas.php 118342 118962 42.4

   Előző óraállás: 118 342 km
   Mostani óraállás: 118 962 km
   Megtett út: 620 km
   Tankolt üzemanyag: 42.4 liter
   Átlagfogyasztás: 6.82 liter/100km
   ```

3. Az elkészített feladatot dockerizálja. Az image neve legyen `monogram/fogyasztas`.

   ```
   $> docker run rcs/fogyasztas 118342 118962 42.4
   Előző óraállás: 118 342 km
   Mostani óraállás: 118 962 km
   Megtett út: 620 km
   Tankolt üzemanyag: 42.4 liter
   Átlagfogyasztás: 6.82 liter/100km
   ```

Kerekítés

1. Hozzon létre egy fájlt `kerekites.php` néven. A szkriptnek legfeljebb két bemenete lehetséges
   - Amennyiben túl sok paramétert kapna, úgy *"Túl sok paraméter!"* hibaüzenettel jelezzen vissza, majd lépjen ki.
   - Az első paraméter a kerekítendő szám lesz. Csak egész szám lehet.
   - A második paraméter a kerekítés módja, megadása opcionális, az alábbi értékek egyike:
     - `fel`, ilyenkor felfelé kerekít, egészre (`felKerekites()`)
     - `le`, ilyenkor lefelé kerekít, egészre (`leKerekites()`)
     - `ft`, ilyenkor az 5 Ft-os kerekítést alkalmazza (`ftKerekites()`).
     - `bankar`, ilyenkor az bankár kerekítést alkalmazza (`bankarKerekites()`)
   - Amennyiben a második paramétert nem adta meg, úgy a matematikai kerekítést alkalmazza (`matematikaiKerekites()`).
   - Egyéb bemenet esetén adjon hibaüzenetet: *"Ismeretlen kerekítési mód!"*, majd lépjen ki
   - Feltételezheti, hogy a felhasználó egész számot adott meg a `ft` kerekítés esetében, míg az összes többi esetében valós számot ad meg.
2. Készítse el az alábbi függvényeket:
   - **`felKerekites($x)`**: a paraméterül kapott `$x` számot felfelé kerekíti.
   - **`leKerekites($x)`**: a paraméterül kapott `$x` számot lefelé kerekíti.
   - **`matematikaiKerekites($x)`** a paraméterül kapott `$x` számot a matematikai kerekítési szabályok alapján kerekíti. 5-től kezdve felfelé, alatta pedig lefelé kerekítsen.
   - **`ftKerekites($x)`**: a paraméterül kapott `$x` számot az 5 Ft-os kerekítés szabályait alkalmazza. Amennyiben a szám 1-re vagy 2-re végződik, úgy lefelé kerekítsen, 8-ra vagy 9-re végződő számokat felfelé kerekítse, a többi számjegy esetén 5-re végződő számot eredményezzen!
   - **`bankarKerekites($x)`**: a paraméterül kapott `$x` számot mindig a hozzá legközelebb eső páros számhoz kerekíti. Például a `2,2` esetén egészre kerekítve `2`-t ad. Míg `3,2` esetén már `4`-et ad eredményül. Bár a matematikai kerekítés szerint 3 lenne, de az nem páros szám. Ami szóba jöhet páros szám az a `2` és a `4`, de ha szmegyenesen vizsgáljuk úgy utóbbihoz áll közelebb.

Kerekítő függvények a PHP dokumentációban:

- http://php.net/manual/en/function.round.php
- http://php.net/manual/en/function.floor.php
- http://php.net/manual/en/function.ceil.php

```
php kerekites.php 1 2 3 4 5
Túl sok paraméter!
```

```
php kerekites.php 1 kerekítsd!
Ismeretlen kerekítési mód!
```

```
php kerekites.php 1352.11 fel
1353
```

```
php kerekites.php 452.99 le
452
```

```
php kerekites.php 8243 ft
8245
```

```
php kerekites.php 11.8 bankar
12
```

```
php kerekites.php 12.8 bankar
12
```

```
php kerekites.php 13.8 bankar
14
```

3. Az elkészített feladatot dockerizálja. Az image neve legyen `monogram/kerekites`.

```
$> docker run rcs/kerekites 11.1 bankar
12
```