1) **Mit ír ki?** 6 pont

A lap jobb szélén van hely a megoldásoknak.

a) Mit ír ki az alábbi python kód?

```
def ritkit(L, atlep = 1):
    i = 0
    while i < len(L):
        del L[i]
        i = i + atlep

L1 = [10, 11, 12, 13, 14, 15]
ritkit(L1)
print L1
L2 = [20, 21, 22, 23, 24, 25]
ritkit(L2, 2)
print L2
```

b) Mit ír ki az alábbi python kód?

```
def ido(tavok, i, sebesseg):
    try:
        return tavok[i] / sebesseg
    except ZeroDivisionError:
        return "Hiba: nulla sebesseg!"
    except:
        return "Hiba: mas hiba!"

print ido([30, 45, 20], 1, 5)
print ido([30, 45, 20], 3, 10)
print ido([30, 45, 20], 0, 0)
```

c) Mit ír ki az alábbi python kód?

```
class Szamlalo(object):
    def __init__(self, nev):
        self.nev = nev
        self.szam = 0

    def novel(self, mennyivel = 1):
        self.szam += mennyivel

sz = Szamlalo("darab")
print sz.szam, sz.nev
sz.novel()
print sz.szam, sz.nev
sz.novel(4)
print sz.szam, sz.nev
```

2) **Kód írás** 9 pont

a) Alap szintaxis (3 pont)

Írjuk le egy osztály teljes definícióját! Az osztályt majd a garanciális tárgyaink garanciáinak tárolására szeretnénk használni. Az osztály neve legyen `Garancia`, és két tagváltozója legyen, `nev`, a tárgy neve, és `lejarat` a grancia lejáratának a dátuma. Elég ha a konstruktor az egyetlen metódus amit definálunk, annak is ugyanez a két paramétere legyen (a `self`-en kívül).

b) Kiegészítés (6 pont)

Írjuk meg a `kell_meg` nevű függvény hiányzó részét! A függvény két paramétert kap:

- `recept` egy recept, szótár formájában, ami minden hozzávaló nevéhez azt rendeli, hogy hány gramm kell belőle
- `van_mar` a meglevő hozzávalók listája, egy lista, aminek minden eleme egy pár, ami azt mutatja, hogy valamilyen hozzávalóból mennyi van már meg. Garantált, hogy a lista csak olyan hozzávalókat tartalmaz, amik a receptben is szerepelnek.

A függvény célja visszaadni hogy melyik hozzávalóból még mennyit kell beszerezni, egy szótár formájában. A visszaadott szótárnak pontosan ugyanazok legyenek a kulcsai, mint a `recept`-nek. **Ha valamelyik hozzávalóból van elég (vagy több mint elég), akkor a szótárban ott 0 szerepeljen!**

A saját kódotokat csak a `for` cikluson belülre írjátok! Az eddigi kód lemásolja a `recept`-et a `kello` nevű változóba, hogy aztán ezt a másolt változatot módosítsa megfelelően, és a végén ebben legyenek a kellő hozzávalók. Ezzel a változóval tér vissza a végén a függvény.

```
def kell_meg(recept, van_mar):
    kello = recept.copy()
    for vam_mar_elem in van_mar:




    return kello
```

Példa felhasználás a következő oldal elején!

Ha jól működik a függvény, ez a példa kód:

```
macaron_teszta = {"mandulaliszt" : 150, "cukor" : 300, "tojasfeherje" : 110}
itthon = [("cukor", 200), ("tojasfeherje", 500)]
print kell_meg(macaron_teszta, itthon)
```

Azt írja ki hogy:

```
{"mandulaliszt" : 150, "cukor" : 100, "tojasfeherje" : 0}
```

3) **Elmélet** 10 pont

a) Alább látható néhány python utasítás, ami után egy (beljebb tabulált) blokknak kell következnie. X-eljük be ezek közül azokat, amelyek ciklust definiálnak (1 pont)

☐ `while` ☐ `class` ☐ `try` ☐ `if`

b) Mi a különbség a *függvény* és a *metódus* között? (1 pont)

c) Írjunk legalább 3 példát *immutable* típusokra. (1 pont)

d) A függvényeknek a paramétereit kétféleképpen lehet átadni, az egyik módszer a *pozíció szerint*, ezt használjuk ha csak egyszerűen sorrendben beírjuk a paramétereket a zárójelbe. Mi a másik módszer neve, és mire jó? (1 pont)

e) Párosítsuk össze a megfelelőket, az alapján, hogy melyikkel illeszkednek együtt a következő mondatba: "A _____ az egyfajta _____." (2 pont)

1. metódus ☐ változó
2. referencia ☐ osztály
3. datetime.date ☐ függvény
4. konstruktor ☐ metódus

f) Mondjunk egy lehetséges okot, ami miatt egy függvény egy *mutable* paraméterét módosíthatja, és egy dolgot amire ilyen függvény írásánál figyelni kell. (2 pont)

g) Milyen speciális hatása van az `__str__()` metódusnak, ha definiálva van egy osztályban? (1 pont)

h) Írjuk le röviden szóban, hogy az *iter()* nevű beépített függvény mire használható! (1 pont)