# Kolokwium II z Wstępu do Informatyki, 2013/2014

1. Telemarketer dzwoni pod numery, które ma na liście. Do każdej osoby z listy dzwoni się tylko raz (aby uniknąć skarg), przy czym liczą się tylko rzeczywiście odbyte rozmowy. Aby skoordynować działania, firma zatrudniająca telemarketerów prowadzi rejestr odbytych rozmów.

   Plik z numerami zawiera w kolejnych linijkach numery telefonów, przykładowo:

   `numery.txt`:

   ```
   501123234
   504142142
   ```

   Pliki z danymi rozmów zawierają: numer telemarketera, numer telefonu, długość rozmowy (0 w przypadku, gdy rozmowa się nie odbyła, tzn. telemarketerowi nie udało się dodzwonić do rozmówcy). Przyjmujemy, że próby są rejestrowane w kolejności chronologicznej. Przykładowy plik:

   `rozmowy.txt`:

   ```
   14 501123234 0
   14 501123234 0
   12 501123234 3
   14 601990880 2
   15 601990880 1
   8 504142142 0
   11 501123234 2
   ```

   (a) (2 pkt.) Napisz funkcję `zadzwonic(f_numery, f_rozmowy)`, dla podanych plików z numerami oraz z rozmowami zwróci listę wszystkich numerów, do których należy jeszcze w danym miesiącu zadzwonić, tzn. tych, pod które nie dodzwonił się skutecznie (tzn. nie porozmawiał przynajmniej minuty) żaden telemarketer. Przykładowo, wynikiem wykonania funkcji `zadzwonic("numery.txt","rozmowy.txt")` powinno być `['504142142']`.

   (b) (4 pkt.) Napisz funkcję `bledy(f_numery, f_rozmowy)`, która pozwoli wyłapać tych telemarketerów, którzy popełnili błędy podczas pracy. Błędem jest w szczególności: próba dzwonienia do osoby, której nie ma na liście oraz próba dzwonienia do osoby, do której już udało się dodzwonić wcześniej jakiemuś telemarketerowi. W przypadku rozpatrywania, czy ktoś popełnił błąd, to, czy telemarketerowi udało się dodzwonić do rozmówcy nie jest brane pod uwagę. Funkcja powinna zwrócić listę numerów telemarketerów, którzy popełnili błędy. Przykładowo, dla `bledy("numery.txt","rozmowy.txt")` wynikiem powinno być `[11, 14, 15]`.

**Rozwiązanie:**

```
def baza_numerow(f_numery):
    f = open(f_numery)
    baza = set([])
    for x in f:
        r = x.split()
    baza.add(r[0])
    f.close()
    return baza

def zadzwonic(f_numery, f_rozmowy):
    baza_n = baza_numerow(f_numery)
    z = set([])
    for k in baza_n
    z.add(k)
    f = open(f_rozmowy)
    for x in f:
        r = x.split()
        if int(r[2]) > 0 and r[1] in z:
```

```
            z.remove(r[1])
    f.close()
    return list(z)

def bledy(f_numery, f_rozmowy):
    baza_n = baza_numerow(f_numery)
    juz_obsluzone = set([])
    bladzacy = set([])
    f = open(f_rozmowy)
    for x in f:
        r = x.split()
        marketer = int(r[0])
        numer = r[1]
        minuty = int(r[2])
        if r[1] not in baza_n or numer in juz_obsluzone:
            bladzacy.add(marketer)
        elif minuty > 0:
            juz_obsluzone.add(numer)
    return list(bladzacy)
```

2. Rozważmy listę list zawierających imiona uczestników kolejnych zebrań koła naukowego (zakładamy, że każde imię jednoznacznie wyznacza jedną osobę). Przykładowa taka lista to np. `[['Jan', 'Maria', 'Krzysztof'], ['Maria', 'Katarzyna'], ['Jan', 'Grzegorz', 'Maria']]`).

   (a) (2 pkt.) Napisz funkcje `kazde(l)`, która zwróci listę osób, które były na każdym z zebrań. Dla naszej przykładowej listy powyżej wynikiem funkcji `kazde` powinno być `['Maria']`.

   (b) (3 pkt.) Napisz funkcję `najczesciej(l, u)`, która dla listy `l` uczestników zebrań oraz dla listy `u` wybranych osób zwróci osobę, która na zebraniach była najczęściej. W przypadku remisu zwracamy dowolną z tych osób, przy czym remis jest także wtedy, kiedy żadna z osób nie była na ani jednym zebraniu.

**Rozwiązanie:**

```
import collections

def spotkania_uczestnik(l):
    uczestnictwa = collections.defaultdict(int)
    for spotkanie in l:
        for uczestnik in spotkanie:
            uczestnictwa[uczestnik] += 1
    return uczestnictwa

def kazde(l):
    ilosc_spotkan = len(l)
    byli = []
    uczestnictwa = spotkania_uczestnik(l)
    for k, v in uczestnictwa.iteritems():
        if v == ilosc_spotkan:
            byli.append(k)
    return byli

def najczesciej(l, u):
    uczestnictwa = spotkania_uczestnik(l)
    przodownik = []
    ile = 0
    for k, v in uczestnictwa.iteritems():
        if k in u:
            if v > ile:
                przodownik = k
                ile = v
  return przodownik
```

3. Co wypiszą następujące programy?

   (a) (2 pkt.)

```
def g(x, z):
    y = x + z[0]
    z[0] = y * 2
```

```
    if x > 3:
        return z
    else:
        return g(x + 1, z)

print g(1, [1])
```

(b) (2 pkt.)

```
v = 2
def h(l):
    global v
    if len(l) > 3:
        return v
    else:
        l.append(v + l[0])
        v += l[0]
        l[0] = v
        return l

print h([2])
```