DS 2 d'informatique - MPSI - Correction

Question 1 – On a :

$$\begin{aligned}\text{RATS}(751) &= S(751 + R(751)) \\ &= S(751 + 157) \\ &= S(908) \\ &= 89\end{aligned} \qquad \begin{aligned}\text{RATS}(6876) &= S(6876 + R(6876)) \\ &= S(6876 + 6786) \\ &= S(13662) \\ &= 12366\end{aligned}$$

Finalement :

$\text{RATS}(751) = 89$ et $\text{RATS}(6876) = 12366$

Question 2 –

```
def egal(L1, L2):
    if len(L1) != len(L1):
        return False
    for i in range(len(L1)):
        if L1[i] != L2[i]:
            return False
    return True
```

Question 3.a – Pour définir `t` à partir de `s`, on peut écrire « `t = s[::-1]` ».

Question 3.b –

```
def revStr(s):
    t = ""
    for c in s:
        t = c + t
    return t
```

Question 4 –

```
def revInt(n):
    s = str(n)
    t = revStr(s)
    return int(t)
```

Question 5.a –

```
def estLC(L):
    for e in L:
        if e < 0 or e > 9:
            return False
    return True
```

Question 5.b – Soit `n = len(L)` la taille de la liste en entrée. On remarque que :

→ Chaque ligne de la fonction s'exécute en temps constant.

→ La boucle `for` fait `n` tours.

Finalement :

La complexité de `estLC` est en $\mathcal{O}$`(n)` avec `n = len(L)`

Question 6.a –

```
def dictC(L):
    assert estLC(L)
    d = {}
    for e in L:
        if e in d:
            d[e] = d[e] + 1
        else:
            d[e] = 1
    return d
```

Question 6.b – Soit `n = len(L)` la taille de la liste en entrée. On remarque que :

→ D'après la question 5.b, l'appel à `estLC` s'exécute en temps $\mathcal{O}(n)$.

→ Mis à part l'appel à `estLC`, toutes les lignes s'exécutent en temps constant.

→ La boucle `for` fait `n` tours de boucle.

Finalement :

La complexité de `dictC` est en $\mathcal{O}(n)$ avec `n = len(L)`

Question 7 –

```
# Solution 1
def triCPT(C):
    d = dictC(C)
    L = []
    i = 1
    while i < 10:
        if (i in d) and (d[i] > 0):
            L.append(i)
            d[i] = d[i] - 1
        else:
            i = i+1
    return L
```

```
# Solution 2
def triCPT(C):
    d = dictC(C)
    L = []
    for i in range(1, 10):
        if i in d:
            for _ in range(d[i]):
                L.append(i)
    return L
```

Question 8 –

```
def int_to_LC(n):
    if n == 0:
        return [0]
    L = []
    while n > 0:
        L.append(n % 10)
        n = n // 10
    return L[::-1]
```

Question 9.a –

```
def estTriee(L):
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return False
    return True
```

Question 9.b –

```
def LC_to_int(L):
    assert estTriee(L)
    n = 0
    for e in L:
        n = 10*n + e
    return n
```

Question 10 –

```
def triInt(n):
    C = int_to_LC(n)
    C = triCPT(C)
    return LC_to_int(C)
```

Question 11 –

```
def RATS(n):
    assert n >= 0
    return triInt(n + revInt(n))
```

Question 12.a – Lorsque $k_0 = 1$:

n	0	1	2	3	4	5
u_n	1	2	4	8	16	77

Question 12.b – Lorsque $k_0 = 4446666$:

n	0	1	2	3
u_n	4446666	1111113	2222244	4446666

On remarque que $u_0 = u_3$ et donc la suite est périodique :

$$\forall n \in \mathbb{N} : \begin{cases} u_{3n} = 4446666 \\ u_{3n+1} = 1111113 \\ u_{3n+2} = 2222244 \end{cases}$$

Question 13 – Montrons par récurrence sur $n \geqslant m$ que $u_{n+T} = u_n$.

Initialisation. Pour $n = m$, l'hypothèse de la question donne :

$$u_{n+T} = u_{m+T} = u_m$$

Hérédité. Soit $n \geqslant m$ tel que $u_{n+T} = u_n$. Montrons que $u_{n+1+T} = u_{n+1}$:

$$u_{n+1+T} = \mathrm{RATS}(u_{n+T}) = \mathrm{RATS}(u_n) = u_{n+1}$$

Conclusion. On vient de montrer que pour tout $n \geqslant m$: $u_{n+T} = u_n$. La suite est donc ultimement périodique de période T (en prenant $n_0 = m$ dans la définition).

Question 14 – Lorsque $k_0 = 27$:

n	0	1	2	3	4	5
u_n	27	99	189	117	288	117

Ainsi, $u_m = u_{m+T}$ avec $m = 3$ et $T = 2$. En vertu de la question 13 :

$$\boxed{P(27) = 2}$$

Question 15 –

```
def periodeV1(k0):
    n_max = 1000
    d = {k0: 0}
    u = k0
    for n in range(1, n_max):
        u = RATS(u)
        if u in d:
            return n - d[u]
        else:
            d[u] = n
    return 0
```

Question 16 –

```
def toutesPeriodes(k0_max):
    L = []
    for k0 in range(0, k0_max+1):
        L.append(periodeV1(k0))
    return L
```

Question 17 –

```
def minPeriode(p):
    assert p == 0 or p == 1 or p == 2 or p == 3 or p == 8 or p == 14 or p == 18
    k0 = 0
    while periodeV1(k0) != p:
        k0 = k0 + 1
    return k0
```

Question 18 – Montrons d'abord que pour tout $m \geqslant 2$, $\mathrm{RATS}(a_m) = b_m$ et $\mathrm{RATS}(b_m) = a_{m+1}$:

$$\begin{aligned}
\mathrm{RATS}(a_m) &= S(a_m + R(a_m)) \\
&= S\Big(12\underbrace{3\ldots3}_{m \text{ fois}}4444 + 4444\underbrace{3\ldots3}_{m \text{ fois}}21\Big) \\
&= S\Big(1233\ \underbrace{3\ldots3}_{m-2 \text{ fois}}\ 4444 + 4444\ \underbrace{3\ldots3}_{m-2 \text{ fois}}\ 3321\Big) \\
&= S\Big(5677\ \underbrace{6\ldots6}_{m-2 \text{ fois}}\ 7765\Big) \\
&= 55\underbrace{6\ldots6}_{m \text{ fois}}7777 \\
&= b_m
\end{aligned}$$

$$\begin{aligned}
\mathrm{RATS}(b_m) &= S(b_m + R(b_m)) \\
&= S\Big(55\underbrace{6\ldots6}_{m \text{ fois}}7777 + 7777\underbrace{6\ldots6}_{m \text{ fois}}55\Big) \\
&= S\Big(5566\ \underbrace{6\ldots6}_{m-2 \text{ fois}}\ 7777 + 7777\ \underbrace{6\ldots6}_{m-2 \text{ fois}}\ 6655\Big) \\
&= S\Big(13344\ \underbrace{3\ldots3}_{m-2 \text{ fois}}\ 4432\Big) \\
&= 12\ \underbrace{3\ldots3}_{m+1 \text{ fois}}\ 4444 \\
&= a_{m+1}
\end{aligned}$$

Conformément à l'énoncé, supposons maintenant qu'il existe $n \in \mathbb{N}$ et $m \geqslant 2$ tels que $u_n = a_m$ (le cas $u_n = b_m$ se traite de manière similaire). On montre alors facilement par récurrence sur $k \in \mathbb{N}$ que :

$$\begin{cases} u_{n+2k} = a_{m+k} \\ u_{n+2k+1} = b_{m+k} \end{cases}$$

Ainsi, pour tout $k \in \mathbb{N}$:

$$\begin{cases} u_{n+2k} = a_{m+k} > 10^{m+5} \\ u_{n+2k+1} = b_{m+k} > 5.10^{m+5} \end{cases}$$

Donc la suite $(u_n)_{n\in\mathbb{N}}$ tend vers $+\infty$ et $P(k_0) = 0$.

Question 19 –

```
# Cette fonction indique s'il existe un m >= 2 tel que u = a_m ou u = b_m
def stop(u):
    L = int_to_LC(u)
    if len(L) < 8:
        return False
    am = [1,2] + [3]*(len(L)-6) + [4,4,4,4]
    if egal(L, am):
        return True
    bm = [5,5] + [6]*(len(L)-6) + [7,7,7,7]
    if egal(L, bm):
        return True
    return False
```

```
def periodeV2(k0):
    d = {k0: 0}
    u = k0
    n = 0
    while not stop(u):
        n = n+1
        u = RATS(u)
        if u in d:
            return n - d[u]
        else:
            d[u] = n
    return 0
```