Written exam

Introduction to Python

Duration: 2 hours

The use of a computer to test your code is allowed.
All your answers should be written and handed in on a piece of paper.
Please indicate clearly the indentations in your code.

Exercise 1 (Computation of a sum)

1. Write a function `sum(N)` that returns the value of the following sum:

$$\underbrace{4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} - \ldots \pm \frac{4}{2N-1}}_{N \text{ terms in total}}$$

   For example, `sum(4)` should return the value of $4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7}$.

2. Compute this sum, taking some high value for N. Which well-known number do you recognize ?

Exercise 2 (Morse code converter)

The Morse code was used to communicate when telegraphy was still in use during the 19th century.
In this exercise, we represent a dot by `0` and a dash by `1`.

Suppose we have already defined a dictionary `morse` that maps characters of the alphabet to strings of `0`'s and `1`'s, representing the Morse code. In other words, suppose we already have the following piece of code:

MORSE CODE

A	•−	M	−−	Y	−•−−
B	−•••	N	−•	Z	−−••
C	−•−•	O	−−−	1	•−−−−
D	−••	P	•−−•	2	••−−−
E	•	Q	−−•−	3	•••−−
F	••−•	R	•−•	4	••••−
G	−−•	S	•••	5	•••••
H	••••	T	−	6	−••••
I	••	U	••−	7	−−•••
J	•−−−	V	•••−	8	−−−••
K	−•−	W	•−−	9	−−−−•
L	•−••	X	−••−	0	−−−−−

```
morse = {"a": "01", "b": "1000", "c": "1010", "d": "100", "e": "0", "f": "0010", "g": "110", "h": "0000", and so on... }
```

---

1. Write a function `morseEncode(S)` that takes some string `S` as an argument and returns its encryption according to the Morse code. It should return `False` whenever unknown characters are encountered in `S`.

   For example `morseEncode("hello")` should return `"0000001000100111"`, and `morseEncode("it's ok")` should return `False` (since the apostrophe and whitespace characters are absent from the dictionary `morse`).

2. Write a function `morseDecode(T)` that takes some encoded string `T` of `0`'s and `1`'s, and returns its decryption according to the Morse code. For example `morseDecode("0000001000100111")` should return `"hello"`.

Exercise 3 (Chained lists)

Chained lists are a recursive data structure which we represent in Python as follows. A *chained list* can be one of two things:

- either the object `None` which represents the empty list.
- either a tuple `(head,tail)` where `head` is of any type, and `tail` is a chained list.

Examples:

- `A = None` defines an empty chained list.
- `B = (3,(1,(2,(1,None))))` defines a chained list containing four elements: the integers 3, 1, 2 and 1. The head of `B` is `3` and the tail of `B` is `(1,(2,(1,None)))`.
- `C = ("hello",None)` defines a chained list containing only one element: the string `"hello"`. The head of `C` is `"hello"` and the tail of `C` is `None` (the empty chained list).

---

1. Write a recursive function `length(L)` that takes a chained list `L` as an argument and returns its number of elements.
2. Write a recursive function `contains(L,x)` that tests whether an element equal to `x` is contained in the chained list `L` (returns a boolean).
3. Write a recursive function `chainedToOrdinary(L)` that takes a chained list `L` as argument, and returns its conversion to an ordinary Python list.

   For example `chainedToOrdinary( (3,(1,(2,(1,None)))) )` should return `[3,1,2,1]`.
4. Write a recursive function `invert(L)` that takes a chained list `L` as an argument, and returns a chained list where the order of the elements of `L` is inverted. At no point you should use ordinary lists.

   For example `invert( (3,(1,(2,(1,None)))) )` should return `(1,(2,(1,(3,None))))`.

   Indication: *Use an accumulator. In other words, the header of your function should be something like:*

```
def invert(L,accumulator=None):
  ...
```