Project 1

Deadline: Sunday, 20 March 2022

In this project you will make use of lists and functions in Python.

Assume you are given a stack of $2k$ cards, $k \in \mathbb{N}_{\geq 2}$. The cards are numbered from top to bottom from 1 to $2k$, so the topmost card is 1.
Now the stack of cards is shuffled in the following way. During an m-shuffling ($m \in \mathbb{N}, m \leq k$) the top m cards are taken off the stack. Then, together with the next m cards, they are used in an alternating fashion to build a new stack, starting with the lower m cards. The remainder of the stack stays as it is.
E.g. the initial $k = 8$ stack (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16) looks like this after one 6-shuffling: (7, 1, 8, 2, 9, 3, 10, 4, 11, 5, 12, 6, 13, 14, 15, 16).
A *complete shuffling* of the stack of $2k$ cards consists of a 1-shuffling, followed by a 2-shuffling, a 3-shuffling, ..., a k-shuffling (in this order).

Write a Python function that, given a number $k \in \mathbb{N}_{\geq 2}$ performs a complete shuffling of the initial stack of $2k$ cards. Upon completion your function should provide the following information:

(a) What are the three topmost cards after the complete shuffling?

(b) After which shuffling was the topmost card from (a) on top for the first time?

(c) How often was the topmost card from (a) on top throughout the shuffling?

Here we use the convention that after the 1-shuffling, the card number 2 was on top once, and all the other cards zero times.

In particular:

- Your function should be defined as `complete_shuffle(k)` and return a list and two integers: `L, s, f`.
- `L` should contain the values of the three topmost cards from (a), `s` should contain the number of the shuffling from (b), and `f` should contain the frequency from (c).

Your function `complete_shuffle` may use all available standard types and methods in Python 3, but must work without importing any packages or modules.

Together with your code you should submit a 1-page report with a brief description of your implementation. By considering an increasing sequence of stack sizes $2k$, and observing the CPU time necessary for your function `complete_shuffle` to finish, formulate an hypothesis on how the CPU time depends on k as k becomes large.

Submission via Moodle

Prepare a Python source file called `project1.py` that contains your function `complete_shuffle`, but nothing else. The docstring of the file `project1.py` must contain your full name. In addition, prepare a report in PDF format, called `project1.pdf`, as outlined above.