CSE101

DUE NOV. 30

23:59 KST

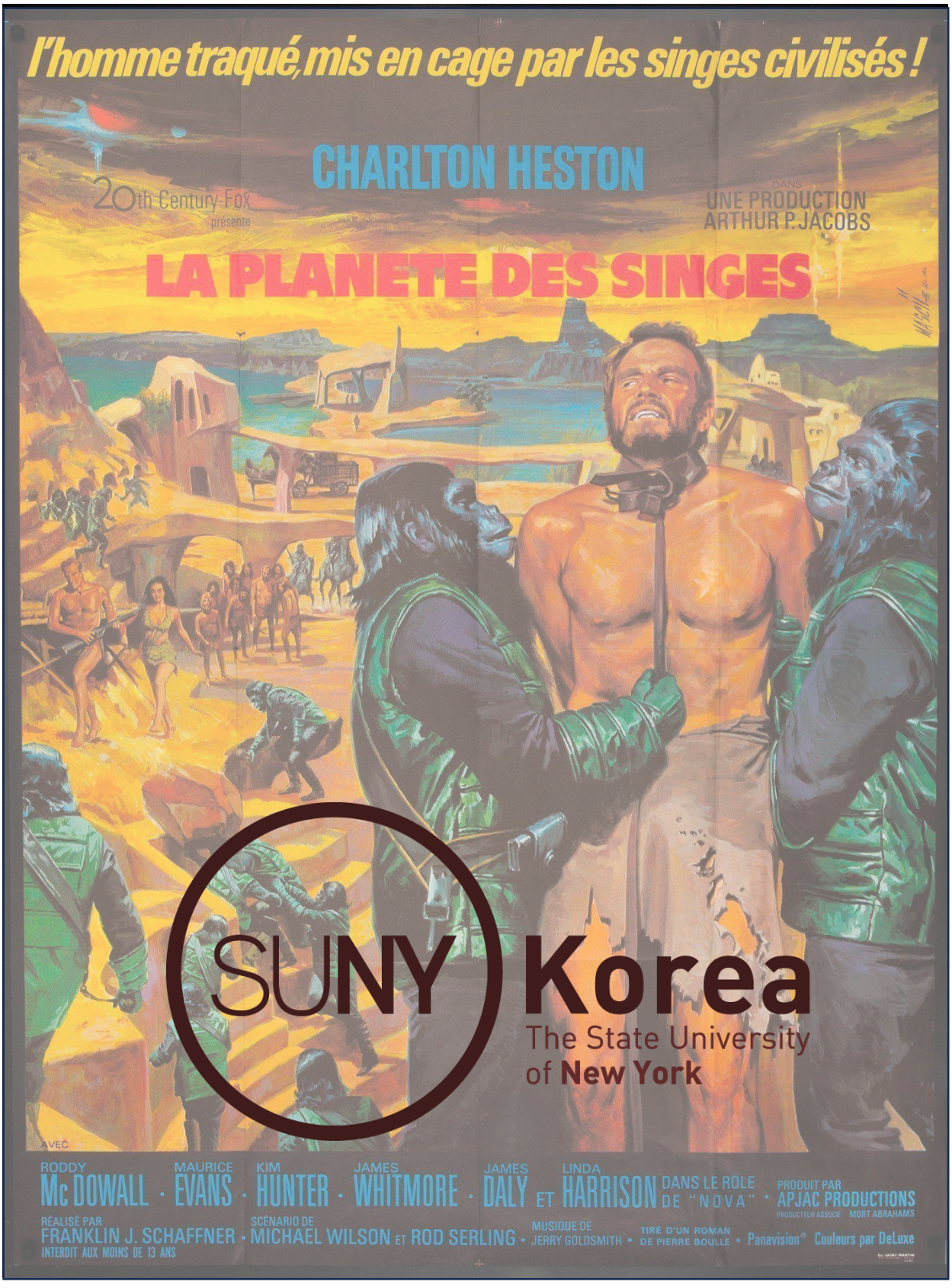
EXTRA CREDIT

PROFESSOR

FRANCOIS

RAMEAU

Sorting



General instructions

- Please add your name and email in the dedicated location at the top of your code
- Do not use any external library except when explicitly requested
- Try to follow the naming convention proposed by PEP-8 seen in class
- Use ONLY what we have already seen in class. If not, you will get ZERO points
- Use meaningful names for your variables and functions
- If you face a problem submitting your code via GitHub, please contact the professor and the TA by email
- Note that the received code will be tested on a classifier to detect the potential usage of Large Language Models. We will also pay particular attention to plagiarism
- Leave comments in your code to explain your code and describe the difficulties you faced

INVITATION LINK

https://classroom.github.com/a/lkoqodby

In this extra credit you will implement two simple sorting algorithms: **Bogo sort** and **Brick sort**.

Exercise 1: Bogo Sort (3 points)

The first algorithm you will implement is Bogo sort. It is known as the **most idiotic** sorting algorithm possible. The principle is very simple: starting from an unsorted list, you will randomly shuffle it until it is sorted! It is the same principle as the Infinite Monkey Theorem, which states that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type any given text, such as the complete works of William Shakespeare. Similarly, if you shuffle a list many, many times, maybe once, by luck, you will end up with a fully sorted list.

Let's implement this algorithm in two steps:

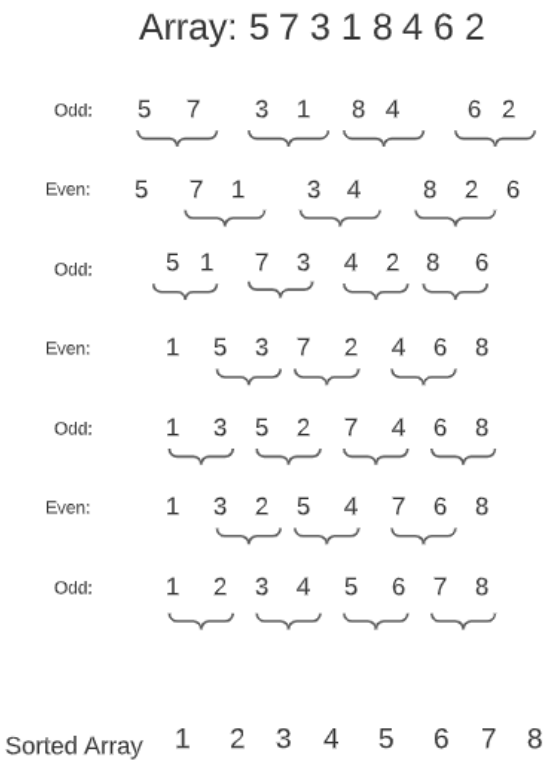
1. You will implement a function `isSorted(lst)` which will return True if the input list is sorted and False if the list is not sorted. You will implement this algorithm to have a complexity ***O(n)*** – 1 point
2. Then implement the bogo sort in the function `bogoSort(lst)`, you will shuffle the list many times only the list is sorted. To shuffle the list, use the "random" python library and the function `random.shuffle(lst)` – 2 points

For fun, you can try to increase the length of the list you are sorting; you will rapidly see that sorting a list of more than 10 elements with bogo sort is a very long task.

Exercise 2: Brick sort (2 points)

The second algorithm you will implement is much more effective, it is the brick sort, also known as the even-odd sort. Do you remember the Bubble sort? Well, it is almost the same, with just a little twist. Similarly, brick sort is a simple comparison-based sorting algorithm. It works by repeatedly comparing pairs of adjacent elements and swapping them if they are in the wrong order. The algorithm separates these comparisons into two phases – one for odd-even indexed pairs and another for even-odd indexed pairs. This process is repeated until the list is sorted. Brick sort is particularly useful for parallel processing systems, as different pairs can be compared and swapped in parallel, enhancing efficiency (we will not implement the multithreaded version for this homework). Implement brick sort in the function BrickSort(lst).

Look at the following figure to better understand its behavior:



You will reuse your function isSorted(lst) to known when to stop swapping elements.