Algorithms Laboratory (CS29203)
Assignment 2: Recursion, Divide & Conquer
Department of CSE, IIT Kharagpur

24th August 2023

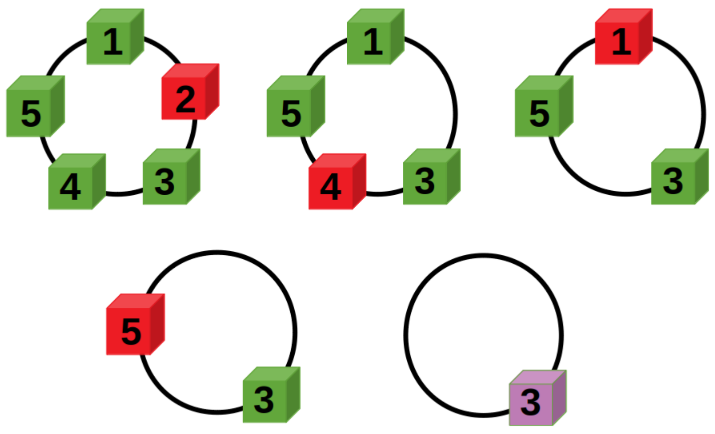
Question-1

You are playing a tye of "treasure hunt" game with your friends. Suppose there are n number of boxes available, where only one box contains the treasure and the rest are empty. The boxes are placed in a circular fashion, numbered from 1 to n in clockwise order. That is, moving from i-th box in clockwise manner by one position will end up to the $(i+1)$-th box. And because of the circular pattern, moving from n-th box in clockwise manner by one position will end up to the first box. Now the goal of the game is that, each of you will choose a number m, and apply it to the circular arrangement of the boxes by some rules (defined below), and end up to a single box after applying the rules. If that box contains the treasure, then the person wins.

We will simulate the rules of the game using computer program and determine which box does one end up with the chosen integer m. The rules of the game are as follows:

- Step 1: Start at the first box
- Step 2: Count the next m boxes in the clockwise direction including the starting box. The counting may wrap around the circle and repeat some boxes more than once.
- Step 3: The last box that is counted, are removed from the arrangement.
- Step 4: If there is > 1 box in the circle, go to Step 2 starting with the box immediately clockwise of the last removed box. Otherwise the last remaining box is the final box, which will be opened for the treasure.

Given the number of boxes n and an integer m, your task is to implement a **recursive** algorithm (no marks for non-recursive solution) that determines the box number after the final iteration. The following is an example with 5 boxes ($n = 5$), and the chosen integer $m = 2$. The final box number is 3. The sequence of steps are shown in the diagram below. Eliminating boxes are shown in red in each step.



Examples:

```
(Input) n = 5, m = 2
(Output) 3
```

```
(Input) n = 6, m = 5
(Output) 1
```

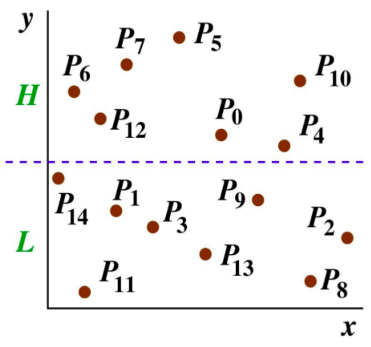
Question-2

You are given a set $\mathcal{S}$ of n points in the 2D plane. For simplicity, assume that no two x-coordinates of the points in $\mathcal{S}$ are the same, and no two y-coordinates of the points in $\mathcal{S}$ are the same. Let P and Q be two points in $\mathcal{S}$. We have one of the four possibilities:

- $x(P) < x(Q)$ and $y(P) < y(Q)$ [we say that Q is dominant over P]
- $x(P) > x(Q)$ and $y(P) > y(Q)$ [P is dominant over Q]
- $x(P) < x(Q)$ and $y(P) > y(Q)$ [neither P nor Q is dominant over the other]
- $x(P) > x(Q)$ and $y(P) < y(Q)$ [neither P nor Q is dominant over the other]

The *dominant index* of a point P in $\mathcal{S}$ is the number of points in $\mathcal{S}$, over which P is dominant. Your task is to compute the dominant index of every point in $\mathcal{S}$ using a **divide and conquer** approach.

There is an obvious quadratic-time algorithm which considers each pair of distinct points in $\mathcal{S}$. However, we are interested in a better solution. You can design a more efficient algorithm by using the divide-and-conquer paradigm. Start by sorting the points in $\mathcal{S}$ with respect to their y-coordinates. Store the sorted list in an array of indices (of points in $\mathcal{S}$). Implement an optimal sorting algorithm for this task. Then use the list sorted with respect to the y-coordinates in order to divide $\mathcal{S}$ into two parts: L (the $n/2$ points of $\mathcal{S}$ with smaller y-coordinates) and H (the $n/2$ points with larger y-coordinates).



In the above figure, the dotted line stands for the division of $\mathcal{S}$ into the two parts L and H. Now recursively compute the dominant indices of the points in L and H. Next, merge the results of these two subproblems to find the final dominant indices of all the points in $\mathcal{S}$. The worst case time complexity of the algorithm should be $O(n \log n)$ (if you consider the 3D case, then the complexity will be $O(n \log^2 n)$, but that is not needed in this assignment). To achieve this running time, your merging step must run in time $O(n \log n)$ or better.

Write a function to implement this **divide-and-conquer** strategy.

Example:

```
(Input) original points: (0.513533,0.663353), (0.419047,0.379945),
(0.238431,0.269107), (0.753736,0.624979), (0.561628,0.204133),
(0.693606,0.244839) (0.189784,0.162833), (0.748571,0.837021),
(0.271007,0.106876) (0.253977,0.136737)

(Output) Dominant indices:  5, 4, 1, 7, 3, 4, 0, 8, 0, 0
```