Programming assignment 1.

Due date: Thursday, February 25, 2021 at 11:59pm

Implement *linearSearch(a,key)* and *binarySearch(a,key)* functions.

Part A. In this part we will calculate the **average-case running time** of each function.

1. Request the user to enter a positive integer, and call it **n**. (*$n = 10^5$*)
2. Generate **n** random integers between *-1000* to *1000* and save them in array **a**.
3. Sort **a** (you can use any sorting algorithm you want.)
4. Pick a random number in **a** and save it in variable called **key**.
5. Call each function separately to search for the key in the given array.
6. To calculate the **average-running time**, you need to have a timer to save the total runtime when repeating step 4 and 5 for **100** times.

(***Note1***: Do not forget to divide the runtime by the number of the times you run step 4-5)

(***Note2***: Remember to choose a different random number each time you go back to step 4.)

Part B. In this part we will calculate the **worst-case running time** of each function.

1. Repeat steps **1 to 3** in part A.
2. Now to have the worst-case scenario, set the value of the key to **5000** to make sure it does not exist in the array.
3. Run each function **ONLY *once*** to calculate the **worst-case running time** when **$n = 10^5$**.
4. Calculate how much time your machine takes to run **one** single line **using only your binary search** function. (***Hint***: what is the time complexity of binary search? How could you use that to calculate the runtime of one line?)
5. Now using the **previous** step, estimate the worst-case running time for each algorithm when **$n=10^{15}$** **(You do not need to run any of binary_search or linear_search functions. Just do a simple calculation in your code!)**.
6. **Explain part 4 and 5 in words.**

Part C. In this part we will change our binary search implementation to solve the below.

Given a sorted array with *n* integers, provide an algorithm with the running time of O(*logn*) that checks if there is an *i* for which *a*[*i*]= *i*.

Example 1: Input: a = [1 1.5 2 5 10 21] ➔ Output: True! (That is because a[2] = 2)
Example 2: Input: a = [1 5 12 17 19 27] ➔ Output: False!