1. Solve the following questions about time complexity.

   a. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n / 2; i > 1; i /= 6) {
        for (j = 2; j <= n; j *= 4) {
                for (k = 0; k <= j; k *= 3) {
                        p = p + n / 2;
                }
        }
}
```

   b. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n; i > 1; i -= 1) {
        for (j = 2; j <= n; j += 1) {
                p = p + n;
                if (p > (n ^ i)) {
                        break;
                }
        }
}
for (i = n / 2; i > 1; i /= 6) {
        for (j = 2; j <= i; j *= 4) {
                for (k = 0; k <= j; k *= 3) {
                        p = p + n / 2;
                }
        }
}
```

Scanned with CamScanner

2. After working with the binary search algorithm, as a CSE221 student you want to explore its strength. You have several ideas in your mind. Try to modify the algorithm to implement your ideas.

   a. What if you want to return the first index of the search key in case there are duplicates. For example: your search key is 13 and there are three 13s in the list.

   b. Now, you also want to return the number of times the key appears in the list. Say, the search key is 54 and it occurs 4 times in indices 6,7,8,9. Then you should return (6,4).

3.

Consider the following list:

Index	0	1	2	3	4	5	6	7
Number	23	2	19	3	7	11	5	13

Will the algorithm given on the right be able to find the search value T=2 for the list above?

If yes, **show** the value of L, R, m in each step of the algorithm. If no, **explain** why not.

```
function binary_search(A, n, T) is
    L := 0
    R := n - 1
    while L ≤ R do
        m := floor((L + R) / 2)
        if A[m] < T then
            L := m + 1
        else if A[m] > T then
            R := m - 1
        else:
            return m
    return unsuccessful
```

Here, *A* denotes the list, and *n* denotes its size.

Scanned with CamScanner

4. You are given an array containing N distinct integers in a wave-like sequence. Meaning, the numbers in the beginning are in ascending order, and after a specific position, they are in descending order. For example: [1, 3, 4, 5, 9, 6, 2, -1]
You have to find the maximum number of this sequence. Can you devise an efficient algorithm such that the time complexity will be less than O(N)?

   i. Present your solution idea as a pseudocode/ python code/ flowchart/ step-by-step instructions/ logical explanation in one-two paragraphs.
   ii. Write the time complexity of your algorithm.

5. Your friend gave you an integer (key) and asked you to find its integer square root without using built-in functions like sqrt() or exponentiation. For example, if the input is 25, the output should be 5 because $5^2=25$. If the square root is not an integer, return the floor value of the actual square root. For example, for an input of 10, the output should be 3.
Your friend's initial approach is to use linear search (time complexity O(n)) as shown below.

```
def LinearSearchToFindSquareRoot(key):
    result = -1
    for i in range(1, key, 1):
        if i * i <= key:
            result = i
    return result
```

   a. Present an $O(\sqrt{n})$ time algorithm with pseudocode/programmable code/step-by-step logical instructions.
   b. Present an $O(log_2 N)$ time algorithm with pseudocode/programmable code/step-by-step logical instructions.

Scanned with CamScanner

6. For different tasks, we often need to search for things. And most of the time, we sort the dataset before performing search operations. Once, you found the following list of numbers.

[2, 5, 1.2, 6.7, 1.7, 9.3, 2.2, 7.7, 0, –4, –5.1, 2, 5, 5.2]

   a. To search an element in an unsorted array, we need O(N) time using linear search. Binary search works in O(logN) time but the array needs to be sorted beforehand which takes at least O(NlogN) time in general. Why would you then sort the array first and then perform binary search instead of just performing linear search?
   b. Can you modify count sort so that it may work with negative integers as well?
   c. Can you modify count sort so that it may work with the given list?
   d. Say, you are working in a system where you need to sort a very big dataset. The memory available to you can barely accommodate the data. You have two options – merge sort & quick sort. Which one should you choose? Why?
   e. Construct an array where quick sort fails to work in O(NlogN) time.

7. Jack loves to play with integers. He created a list of n integers where the even indices hold numbers in decreasing order and the odd indices hold numbers in increasing order. For example, this is a list of n=8 integers Jack made.

Index	0	1	2	3	4	5	6	7
Number	23	2	19	3	7	11	5	13

[Explanation:
The indices 1, 3, 5, and 7 have numbers 2, 3, 11, and 13 in increasing order.
The indices 0, 2, 4, and 6 have numbers 23, 19, 7 and 5 in decreasing order.]

   a. Present linear time algorithm to sort the elements either in ascending or descending order with pseudocode/programmable code/step-by-step logical instructions.

Scanned with CamScanner