**ASSIGNMENT – 04**

**NAME: Anupam Anant**

**Question 1**

**In the Binary Search algorithm, it is suggested to calculate the mid as**

**beg + (end - beg) / 2 instead of (beg + end) / 2. Why is it so?**

**Ans:**

In the Binary Search algorithm, it is suggested to calculate the mid as

beg + (end - beg) / 2 because there is no danger of overflow.

Eg.

Integer range is -2,147,483,648 to 2,147,483,647. If we are searching in an array of size 2,000,000,000 and the element searched for is located at index 1,999,999,999. When we search in the upper half of array, beg=1,000,000,001 and end=2,000,000,000. If mid is calculated as (low+high)/2, low+high = 3,000,000,001; which exceeds the range of int, resulting in overflow errors. But mid calculated as beg + (end-beg) = 1,000,000,001 + 999,999,999 = 2,000,000,000; which fits in the integer range.

**Question 2**

**Write the algorithm/function for Ternary Search.**

**Ans:**

Like linear search and binary search, ternary search is a searching technique that is used to determine the position of a specific value in an array. In binary search, the sorted array is divided into two parts while in ternary search, it is divided into 3 parts and then we determine in which part the element exists.

Ternary search is a divide-and-conquer algorithm. It is mandatory for the array to be sorted before we begin the search. In this search, after each iteration it neglects ⅓ part of the array and repeats the same operations on the remaining ⅔.

int ternarySearch(int l, int r, int key, int ar[])

{

    while (r >= l)

{

        int mid1 = l + (r - l) / 3;

        int mid2 = r - (r - l) / 3;

        if (ar[mid1] == key) {

            return mid1;

        }

        if (ar[mid2] == key) {

            return mid2;

        }

        if (key < ar[mid1]) {

            r = mid1 - 1;

        }

        else if (key > ar[mid2]) {

            l = mid2 + 1;

        }

        else {

            l = mid1 + 1;

            r = mid2 - 1;

        }

    }

    return -1;

}

Time Complexity: O(log3 n)