**Assignment 4**

**Q1:-** In the binary search algorithm, it is suggested to calculated the mid as beg + (beg - end)/2 instead of (beg + end)/2. Why is it so?

**Answer:--** Binary search is the search technique which works efficiently on the sorted lists. In order to search an element into some list by using binary search technique, we must ensure that the list is sorted. Binary search follows divide and conquer approach in which, the list is divided into two halves and the item is compared with the middle element of the list. If the match is found then, the location of middle element is returned otherwise, we search into either of the halves depending upon the result produced through the match.

BINARY\_SEARCH(A, lower\_bound, upper\_bound, VAL)

* **Step 1:** [INITIALIZE] SET BEG = lower\_bound  
  END = upper\_bound, POS = - 1
* **Step 2:** Repeat Steps 3 and 4 while BEG <=END
* **Step 3:** SET MID = (BEG + END)/2
* **Step 4:** IF A[MID] = VAL  
  SET POS = MID  
  PRINT POS  
  Go to Step 6  
  ELSE IF A[MID] > VAL  
  SET END = MID - 1  
  ELSE  
  SET BEG = MID + 1  
  [END OF IF]  
  [END OF LOOP]
* **Step 5:** IF POS = -1  
  PRINT "VALUE IS NOT PRESENT IN THE ARRAY"  
  [END OF IF]
* **Step 6:** EXIT

**Q2 :-- write a algorithm for ternary search?**

**Answer:-- Ternary search** is a [divide and conquer algorithm](http://www.geeksforgeeks.org/divide-and-conquer-introduction/) that can be used to find an element in an [array](https://www.geeksforgeeks.org/array-data-structure/). It is similar to [binary search](http://www.geeksforgeeks.org/binary-search/) where we divide the array into two parts but in this algorithm, we divide the given array into three parts and determine which has the searched element. We can divide the array into three parts by taking mid1 and mid2. Initially, l and r will be equal to 0 and n-1 respectively, where n is the length of the array.

**Steps to perform Ternary Search:**

1. First, we compare the key with the element at mid1. If found equal, we return mid1.
2. If not, then we compare the key with the element at mid2. If found equal, we return mid2.
3. If not, then we check whether the key is less than the element at mid1. If yes, then recur to the first part.
4. If not, then we check whether the key is greater than the element at mid2. If yes, then recur to the third part.
5. If not, then we recur to the second (middle) part.