DAY-4 ASSIGNMENT | 28th December, 2020

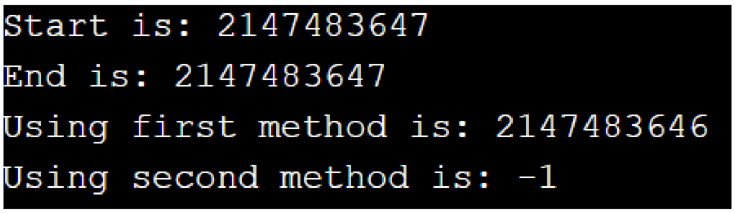
1. Problem Statement:
In the Binary Search algorithm, it is suggested to calculate the mid as beg + (end - beg) / 2 instead of (beg + end) / 2. Why is it so?

Solution:
The first method i.e., beg+(end-beg)/2 is the best way to find the middle index of the array. The drawback of the second one is that if both the start and end values are equal to INT_MAX, it will cause an overflow.

Demonstration using a program:

```
#include<stdio.h>
#include<limits.h>
int main()
{
   int start = INT_MAX, end = INT_MAX;
   printf("Start is: %d\n",start);
   printf("End is: %d\n",end);
   int mid1 = start + (end +  start)/2;
   printf("Using first method is: %d\n",mid1);
   int mid2 = (start + end) /2;
   printf("Using second method is: %d",mid2);
   return 0;
}
```

Output:



```
Start is: 2147483647
End is: 2147483647
Using first method is: 2147483646
Using second method is: -1
```

Explanation of the output:
Here, we can observe that using the first method we get the correct output and the second method fails(-1), it can cause segmentation fault because of invalid index of array. It is because pointer addition is not possible.

2.Problem Statement:
Write the algorithm/function for Ternary Search.

Solution:
Like linear and binary search, we also have ternary search that is used to find out the position of a specific element.
In binary search, the sorted array is divided into two parts while in ternary search, it is divided into 3 parts and then you determine in which part the element exists.

Ternary search is also a type of divide and conquer method. It is mandatory for the array to be sorted before you begin the search, after each iteration it neglects ⅓, part of the array and repeats the same operations on the remaining ⅔.

Function:

```
int ternary_search(int l,int r, int x)
{
    if(r>=l)
    {
        int mid1 = l + (r-l)/3;
        int mid2 = r -  (r-l)/3;
        if(ar[mid1] == x)
            return mid1;
        if(ar[mid2] == x)
            return mid2;
        if(x<ar[mid1])
            return ternary_search(l,mid1-1,x);
        else if(x>ar[mid2])
            return ternary_search(mid2+1,r,x);
        else
            return ternary_search(mid1+1,mid2-1,x);
    }
    return -1;
}
```