Assignment No. 4 Data Structures and Algorithms

**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?**

**Answer:**

1)First of all, **beg + (end - beg) / 2**  works even if you are using pointers, as long as  **end-beg**  doesn't overflow.

2) Second of all, beg + (end -beg) / 2 won't overflow if beg and end are large positive numbers. With signed operands, overflow is undefined. (Note that end – beg may overflow, but only if beg < 0 or end <0).

Or with unsigned arithmetic, overflow is defined but gives you the wrong answer. However, for unsigned operands, **beg + (end - beg) / 2**   will never overflow as long as  end >= beg.

**Question 2:**

**Write the algorithm/function 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 key (searched element).

#include <stdio.h>

// Function to perform Ternary Search

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

{

if (r >= I) {

// Find the mid1 and mid2

int mid1 = l + (r - l) / 3;

int mid2 = r - (r - l) / 3;

// Check if key is present at any mid

if (ar[mid1] == key) {

return mid1;

}

if (ar[mid2] == key) {

return mid2;

}

if (key < ar[mid1]) {

// The key lies in between l and mid1

return ternarySearch(l, mid1 - 1, key, ar);

}

else if (key > ar[mid2]) {

// The key lies in between mid2 and r

return ternarySearch(mid2 + 1, r, key, ar);

}

else {

// The key lies in between mid1 and mid2

return ternarySearch(mid1 + 1, mid2 - 1, key, ar);

}

}

return -1;

}

int main(){

int l, r, p, key;

int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

l = 0;

r = 9;

key = 5;

p = ternarySearch(l, r, key, ar);

printf("Index of %d is %d\n", key, p);

key = 50;

// Search the key using ternarySearch

p = ternarySearch(l, r, key, ar);

printf("Index of %d is %d", key, p);

}