Day 4 assignment

Ans1.

If we consider the two lines in a more generic setting, not related to binary search, the following observations can be made:

You are correct that the problem the second form tries to avoid is overflow, attempting to represent a number that is larger than the maximum representable number.

There is no restriction on how large the individual numbers beg and end are, so potentially they can both be larger than half of the maximum representable number. Adding them means that the intermediate result (beg+end) can overflow.

The second solution seems to eliminate the risk of overflowing, but introduces another one. If the values are signed values, their difference can again overflow (or underflow, depending on their signs). Unsigned values have no problem.

Ans2. Ternary Search

Ternary search is a divide and conqur algorithm that can be used to find an element in an array. It is similar to 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).

Algorithm for 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.

Function for ternary search

#include <stdio.h>

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;

}