1. There's no guarantee that beg+end is representable; but in the second case the intermediate values, as well as the expected result, are no larger than end, so there is no danger of overflow. So the second one [beg+(end-beg)/2] can avoid overflow.

#include <bits/stdc++.h>

using namespace std;

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

{

    if (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])

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

        else if (key > ar[mid2])

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

        else

            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);

    cout << "Index of " << key;

         << " is " << p << endl;

}