```
/*
Nhan Vo
SID#017771388
CECS 328-EC 1-Lab 1
*/
public class Ec1 {
   public static void main(String[] args){
       Ec1 ec=new Ec1();
       System.out.println("-----Problem 1 testing-----");
       int[] a={0,0,0,1,1};
       int[] b={0,0,0,0,0,1,1,1};
       System.out.println(ec.findSplit(a));
       System.out.println(ec.findSplit(b)+"\n");
       System.out.println("-----Problem 2 testing-----");
       int[] c={0,2,10,26,68};
       int[] d={1,11,18,20,41};
       System.out.println(ec.findMedian2lists(c,d)+"\n");
       int[] e={5,6,14,26};
       int[] f={3,41,88,100};
       System.out.println(ec.findMedian2lists(e,f)+"\n");
       int[] g={5,10};
       int[] h={2,41};
       System.out.println(ec.findMedian2lists(g,h)+"\n");


   }
   /*

   Problem 1:
   Function findSplit that find the index that split 0 and 1
   How it works:
       First, we implement the normal binary search algorithm.
   Then, we check the variable in array a at index mid.

   If a[mid] is different from a[mid-1], mean that a[mid-1] is 0 and a[mid] is
1 due to a is sorted array.

   If a[mid] = 1 and a[mid]==a[mid-1], mean we find a pair of 1. In that case,
we need to reduce the end point. Hence, end=mid-1

   If a[mid] = 0 and a[mid]< a[mid+1], mean we also find a pair of 0 and 1 but
instead of 1 at mid, we have 1 at mid+1. Hence, return mid+1

   If a[mid] = 0 and a[mid]=a[mid+1], mean we find a pair of 0 and we need to
increase the start point. Hence, start= mid+1

   The function will run until it find the result or start point value= end
point value, which mean no answer. In that case, it will return -1
```

How I came up with the idea:

First, I just do a normal binary search for the provided array in the example. Then I realized that since it is a binary array,

the split can be detected if we have a different pair of number like a pair of 0 and 1 or 1 and 0. Simultaneously, if we have a similar

pair, for example a pair of 0, we know that we need to continue moving to the right side (increase starting point). And if we have a

similar pair of 1, we know that we need to continue moving to the left (decrease end point).

```
*/
public int findSplit(int[] a){
    int start=0;
    int end=a.length;
    while(start<end){
        int mid=(start+end)/2;

        if(a[mid-1]!=a[mid]){
            return mid;
        }
        else if(a[mid]==0&&a[mid]<a[mid+1]){
            return mid+1;
        }
        else if(a[mid]==1&&a[mid]==a[mid-1]){
            end=mid-1;
        }
        else if(a[mid]==0&&a[mid]==a[mid+1]){
            start=mid+1;
        }

    }
    return -1;
}

/*
```

Problem 2:

Function findMedian2lists is use to find the median of two sorted lists in O(log(n)) time

How it works:

First, we apply binary search for one array. Since two arrays have the same length, the other length will be the original length subtract for the midpoint of the first array.

We want to draw a line across two arrays so that it can divide the two list into two part (left and right) that any number in the left part will less than any number in the right part.

If the mid point of the first array does not satisfy this condition, we continue to update the starting point for the first array. Keep in mind that we will never change the end point which is the length.

Then we continue increment the mid point by taking (start point+ end point)/2

If the mid point satisfies the previous condition, we calculate the result by taking the min of the value of the mid point of the first array and the value of one number before the mid point of the second array
then add it to the min of the value of one number after the mid point of the first array and the mid point of the second array, together divided by 2.

How I came up with the idea:
I started with example#3 since it is the easiest example with a calculation. I see the median formula Median = {max(a1[0], a2[0]) + min(a1[1], a2[1])}/2.
My first thought was that the formula will be something like Median = {max(a1[mid point], a2[mid point]) + min(a1[mid point-1], a2[mid point-1])}/2. I also saw that
if I draw a vertical line in the middle of the two array, I will have one all number in the left will always be less than number in the right, regardless of the comparison.
I started doing the second example on paper with the similar thought as the 3rd example. However, I soon encountered obstacle when I try to increment the mid point of the two array similar to each other
I realized that even with a horizontal line, I could not get the same answer as the output. My first answer for the 2nd example is (5+6)/2=5.5, not 20 like in the output. So, I think back about the way I
increment the two mid points. Instead of drawing a horizontal line, this time, I draw multiple diagonal, vertical line between two number in two arrays. And I saw that, when I draw a diagonal line between
14 and 41. I have something similar to the formula that I had came up with Median = {max(a1[mid point], a2[mid point]) + min(a1[mid point-1], a2[mid point-1])}/2 but the order is different. So I change the
increment for 2nd array= the length - mid point of the first array, since the separation line is a diagonal one like y=x. As the mid point of the first array move closer to the end, the mid point of the second
array will move closer to 0. Then I change my formula into (Math.max(a[aMid],b[bMid-1])+Math.min(a[aMid+1],b[bMid]))/2. And I use the same formula and increment condition to redo example#1 on paper.
I glad it works. Yay!!!!!!
*/

```
public double findMedian2lists(int[] a, int[] b){
    int length=a.length-1;
    double abMed=-1;
    int start=0;
    int end=length;
    int aMid=0;
```

```
        int bMid=0;
        while(start<end){
            aMid=(start+end)/2;
            bMid=end-aMid;

            if(Math.max(a[aMid],b[bMid-1])<=Math.min(a[aMid+1],b[bMid])){

abMed=(double)(Math.max(a[aMid],b[bMid-1])+Math.min(a[aMid+1],b[bMid]))/2;
                return abMed;
            }
            else if(Math.max(a[aMid],b[bMid-1])>Math.min(a[aMid+1],b[bMid])){
                start=aMid;
            }

        }

        return abMed;
    }

}
```