**Exercise 1.4.8** Write a program to determine the number pairs of values in an input file that are equal. If your first try is quadratic, think again and use Arrays.sort() to develop a linearithmic solution.

**Solution:**

- Since we want a linearithmic solution, like we can use Binary search due to $log_2 N$ complexity while iterating through the array to search for the next element that isn't the same as previous element..
- If the array is not sorted, then sort the array using $Array.sort()$
- Go through the array of elements.
- Use binary search to find first and last instance of an element.
- subtract the indices and then add 1 since our indices start at index 0.
- Check for the next element in array, if it is an element we already saw previously, then ignore it. If found a new element, then again repeat above steps using BinSearch to find first and last occurences. Basically, do this for each unique element.
- ie; 1,2,2,2,2,3

Pseudocode below

```
//PseudoCode: Count number of equal pairs
//Anando Zaman
function FindFirst(nums[],key,low,high): //first index of key in nums

    occurence = -1 //since we don't know if it occurs at all
    while(high>low): //iterative binsearch

        mid = low + (high-low)/2

        if(key == nums[mid]):
            occurence = mid

            //we want first occurence that is smallest
            //so adjust high to get min high by checking LHS,
            //which contains smaller values or smallest equal occurence.
            //We could also just keep checking backwards but takes O(n) complexity
            high = mid-1

        else if(key > nums[mid]):
            low = mid + 1 //check right half, RHS

        else:
            high = mid - 1 //check LHS

    return occurence

function FindLast(nums[],key,low,high):

    occurence = -1 //since we don't know if it occurs at all
    while(high>low): //iterative binsearch

        mid = low + (high-low)/2

        if(key == nums[mid]):
            occurence = mid

            //we want last occurence of the value
            //so check right-hand side subarray, RHS
            low = mid+1

        else if(key > nums[mid]):
            low = mid + 1 //check right half, RHS

        else:
            high = mid - 1 //check LHS

    return occurence
```

```
function Occurences(nums[], key, low, high):
    First_occurence = findFirst(nums,key,low,high)
    if(First_occurence > -1): //returned a valid index
        return (First_occurence - findLast(nums,key,low,high) + 1)

function Pairs (nums []):
    Array.sort(nums)

    int previous_elm = -1;
    int count = 0;

    for i in range(nums.length):

        //to avoid reading and looking for duplicates if already seen previously
        if(previous_elm != nums[i]){
            previous_elm = nums[i]
            count += Occurences(nums,nums[i],i+1, nums.length-1)
        }

    return count
```