Xinghan Zhao
CSE331 HW0
Student ID: 1560903

```
public static void rearrange(int[] b){
    int n = b.length;
    int front = 0;
    for(int i = 0; i < n; i++){
        if(b[i] < 0){
            if(i != front){
                swap(b[i], b[front]);
            }
            front++;
        }
    }
    for(int i = front; i < n; i++){
        if(b[i] == 0){
            if(i != front){
                swap(b[i], b[front]);
            }
            front++;
        }
    }
}
```

For this assignment, I used about 90 minutes. During the first 60 minutes, I tried to "implement it one pass through the array", by setting two pointers pointing both the front and the back of this array, and checking numbers from both ends together. However, I found this was difficult to write since it would have a huge number of "while" and "if else" when I tried to put my idea down to the paper, so I gave up and chose this easier solution.

So, in my final solution, the method "rearrange" will take the array b, which will at least have one number (n>=1), as the only parameter, get its length, and use two for loops to check the numbers. There is only one pointer "front" pointing to the front of uncompleted subarray.

In the first for loop, it will go through the entire array to find numbers that less than 0. If it find one, it will swap it with the number located at the "front", then the pointer "front" will be moved one position forward, meaning that former numbers are all less than 0. The reason we need a "if(i != front)" is to make sure it will not do useless job to swap the number with itself.

And in the second for loop, it does similar job, but begin with the front of unchecked subarray, and move all 0s right after the negative numbers. After that, all positive numbers will be left in the back of the array and no need to check them again.