(9) Find longest subarray whose sum equals target number.
- use sliding window, if sum > k, increment i, if (sum == k), update start and end indexes, and inc i,

```
void longestSubarraySum(int[] a, int k) {
    int i=0, j=0, start=0, end=0, length=0, sum=0;
    while(j < a.length && i <= j){
        sum += a[j];
        if(sum > k){ sum -= a[i]; i++}
        else if(sum == k && (j-i+1) > length){
            length = j-i+1;
            start = i; end = j;
            sum -= a[i]; i++;
        }
        j++;
    }
    s.o.p(start+" "+end);
}
```

So that we can find more subarrays

(10) Find all duplicates in array where values are 1 to n-1.
- Since values are between 1 to n-1, for each element at a[i], change a[a[i]] = a[a[i]]*(-1); if it is >0 else print i.

```
for(int i=0; i<a.length; i++){
    ~~if(a[i]<0) s.o.p(i);~~ if(a[Math.abs(a[i])] < 0 { s.o.p(a[i]);
    ~~else a[a[i]] = -1 * a[a[i]];~~ else a[Math.abs(a[i])] = -a[Math.abs(a[i])];
}
```