Two Sum Closest

```
public class Solution {
    /**
     * @param nums an integer array
     * @param target an integer
     * @return the difference between the sum and the target
     */
    public int twoSumCloset(int[] nums, int target) {
        if (nums == null || nums.length < 2) {
            return -1;
        }

        if (nums.length == 2) {
            return target - nums[0] - nums[1];
        }
        Arrays.sort(nums);
        int pl = 0;
        int pr = nums.length - 1;

        int minDiff = Integer.MAX_VALUE;
        while (pl < pr) {
            int sum = nums[pl] + nums[pr];
            int diff = Math.abs(sum - target);
            if (diff == 0) {
                return 0;
            }
            if (diff < minDiff ) {
                minDiff = diff;
            }
            if (sum > target) {
                pr--;
            } else {
                pl++;
            }
        }
        return minDiff;
    }
}
```

Window Minimum。具体题目如下：
给了一个ArrayList：4, 2, 12, 11, -5，窗口size为2，返回的ArrayList为：2, 2, 11, -5。这里窗口size是一个参数。

```
public class Solution {
    public int[] maxSlidingWindow(int[] nums, int k) {

        if (nums == null || k <= 0) {
            return new int[0];
        }
        int n = nums.length;
        int[] r = new int[n-k+1];
        int ri = 0;
        // store index
        Deque<Integer> q = new ArrayDeque<>();
        for (int i = 0; i < nums.length; i++) {
            // remove numbers out of range k
            while (!q.isEmpty() && q.peek() < i - k + 1) {
                q.poll();
            }
            // remove larger numbers in k range as they are useless
            while (!q.isEmpty() && nums[q.peekLast()] > nums[i]) {
                q.pollLast();
            }
            // q contains index... r contains content
            q.offer(i);
            if (i >= k - 1) {
                r[ri++] = nums[q.peek()];
            }
        }
        return r;

    }
}
```

High five

```
public class Solution {
    /**
     * @param results a list of <student_id, score>
     * @return find the average of 5 highest scores for each person
     * Map<Integer, Double> (student_id, average_score)
     */
    public Map<Integer, Double> highFive(Record[] results) {
        // Write your code here
        Map<Integer, PriorityQueue<Integer>> hashMap = new HashMap<>();
        Map<Integer, Double> result = new HashMap<>();

        for(Record record : results){
            if(hashMap.containsKey(record.id)){
                PriorityQueue<Integer> temp = hashMap.get(record.id);
                temp.add(record.score);
                if(temp.size() > 5){
                    temp.poll();
                }
            }else{
                PriorityQueue<Integer> toAdd = new PriorityQueue<>();
                toAdd.add(record.score);
                hashMap.put(record.id, toAdd);
            }
        }
        for (Map.Entry<Integer, PriorityQueue<Integer>> entry : hashMap.entrySet()){
            int index = entry.getKey();
            PriorityQueue<Integer> temp = entry.getValue();
            double score = 0;
            while(temp.size()>0){
                score += temp.poll();
            }
            score /= 5;
            result.put(index, score);
        }
        return result;
    }
}
```

Copy list with random pointer

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

```
    public RandomListNode copyRandomList(RandomListNode head) {
        if(head == null) return null;

        Map<RandomListNode, RandomListNode> map = new HashMap<>();
        RandomListNode cur = head;

        while(cur != null){
            RandomListNode toAdd = new RandomListNode(cur.label);
            map.put(cur,toAdd);
            cur = cur.next;
        }

        cur = head;
        while(cur != null){
            map.get(cur).next = map.get(cur.next);
            map.get(cur).random = map.get(cur.random);
            cur = cur.next;
        }

        return map.get(head);
    }
```