Programming:  
1. right rotation.  
Code:  
        public static int rightRotate(String word1, String word2) {. visit 1point3acres.com for more.  
                if (word1 == null || word2 == null || word1.length() == 0 || word2.length() == 0 || word1.length() != word2.length()) {  
                        return -1;  
                }  
                String str = word1 + word1;.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
                return str.indexOf(word2) != -1 ? 1 : -1;  
        }  
  
        18/18 passed  
2. grey code. 没用地里大神的x & (x - 1)这个做法。  
        public static int greyCode(byte element1, byte element2) {  
                byte res = (byte) (element1 ^ element2);  
                for (int i = 0; i <= 7; i++) {  
                        byte temp = (byte)(1 << i);.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�  
                        if (temp == res) {. From 1point 3acres bbs  
                                return 1;  
                        }  
                }  
                Sy[stem](http://www.1point3acres.com/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%9C%9F%E5%AE%89%E5%85%A8%E9%83%A82012%E5%B9%B4%E5%BA%A6%E6%9B%B4%E6%96%B0stem%E4%B8%93%E4%B8%9A%E5%90%8D%E5%8D%95-%E7%9C%8B%E7%9C%8B%E4%BD%A0%E7%9A%84%E4%B8%93%E4%B8%9A/" \t "_blank).out.println("No");  
                return 0;  
        }  
  
        16/16 passed.

刚刚做完A家的ONLINE测试，两道代码题很简单，去元音和BYTE GRAY CODE；  
去元音：.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�  
//string 是题目给的字符串  
StringBuffer sb = new StringBuffer();  
String v = "aeiouAEIOU";  
for(int i = 0; i < string.length(); i++){  
if(v.indexOf(string.charAt(i)) > -1) continue; 鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�.   
sb.append(string.charAt(i));  
}  
return sb.toStirng();  
一次通过所有TEST(本渣做的时候少写了右半边括号 还少些了一个分号。。。所以不是不一遍过的)  
  
byte:  
byte这道题有好多同学都说最后一两个TEST通过不了，我是一遍过的 我觉得可能跟判断亦或之后的数值类型有关：  
//term1和term2是题目给的两个BYTE  
byte x = (byte)(term1 ^ term2);  
int total = 0;  
while(x != 0){  
x = (byte) (x & (x - 1));.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
total++;  
}  
if(total == 1) return 1; else return 0;  
一遍过；

valid parenthe:

import java.util.Stack;

public class isValid {

public boolean isValidParentheses(String s) {

if (s == null || s.length() == 0) return true;

Stack<Character> stack = new Stack<Character>();

for (int i = 0; i < s.length(); i++) {

if (stack.empty()) stack.push(s.charAt(i));

else if (s.charAt(i) - stack.peek() == 1 || s.charAt(i) - stack.peek() == 2) stack.pop();

else stack.push(s.charAt(i));

}

return stack.empty();

}

}

package longestPalindrome;

public class Solution {

public String longestPalindrome(String s) {

String T = preProcess(s);

int n = T.length();

int[] p = new int[n];

int center = 0, right = 0;

for (int i = 1; i < n - 1; i++) {

int j = 2 \* center - i; //j and i are symmetric around center

p[i] = (right > i) ? Math.min(right - i, p[j]) : 0;

// Expand palindrome centered at i

while (T.charAt(i + 1 + p[i]) == T.charAt(i - 1 - p[i]))

p[i]++;

// If palindrome centered at i expand past right,

// then adjust center based on expand palindrome

if (i + p[i] > right) {

center = i;

right = i + p[i];

}

}

// Find the longest palindrome

int maxLength = 0, centerIndex = 0;

for (int i = 1; i < n - 1; i++) {

if (p[i] > maxLength) {

maxLength = p[i];

centerIndex = i;

}

}

centerIndex = (centerIndex - 1 - maxLength) / 2;

return s.substring(centerIndex, centerIndex + maxLength);

}

// preProcess the original string s.

// For example, s = "abcdefg", then the rvalue = "^#a#b#c#d#e#f#g#$"

private String preProcess(String s) {

if (s == null || s.length() == 0) return "^$";

StringBuilder rvalue = new StringBuilder("^");

for (int i = 0; i < s.length(); i++)

rvalue.append("#").append(s.substring(i, i+1));

rvalue.append("#$");

return rvalue.toString();

}

}

package mergeTwoLists;

class ListNode {

int val;

ListNode next;

ListNode(int x) { val = x; }

}

public class Solution {

public ListNode mergeTwoLists(ListNode l1, ListNode l2) {

ListNode head = new ListNode(0);

ListNode cur = head;

while (l1 != null && l2 != null) {

if (l1.val < l2.val) {

cur.next = l1;

l1 = l1.next;

}

else {

cur.next = l2;

l2 = l2.next;

}

cur = cur.next;

}

cur.next = (l1 != null) ? l1 : l2;

return head.next;

}

}

half list:

public static ListNode reverseSecondHalfList(ListNode head) {

if (head == null || head.next == null) return head;

ListNode fast = head;

ListNode slow = head;

while (fast.next != null && fast.next.next != null) {

fast = fast.next.next;

slow = slow.next;

}

ListNode pre = slow.next;

ListNode cur = pre.next;

while (cur != null) {

pre.next = cur.next;

cur.next = slow.next;

slow.next = cur;

cur = pre.next;

}

return head;

}

subtree:

public class Subtree {

public boolean isSubTree(TreeNode T1, TreeNode T2) {

if (T2 == null) return true;

if (T1 == null) return false;

return (isSameTree(T1,T2) || isSubTree(T1.left, T2) || isSubTree(T1.right, T2));

}

public boolean isSameTree(TreeNode T1, TreeNode T2) {

if (T1 == null && T2 == null) return true;

if (T1 == null || T2 == null) return false;

if (T1.val != T2.val) return false;

return (isSameTree(T1.left, T2.left) && isSameTree(T1.right, T2.right));

}

}

two sum:

import java.util.HashMap;

import java.util.Map;

public class Solution {

public static int TwoSumCount(int[] nums, int target) {

if (nums == null || nums.length < 2) return 0;

Map<Integer, Integer> map = new HashMap<Integer, Integer>();

int count = 0;

for (int i = 0; i < nums.length; i++) {

if (map.containsKey(target - nums[i]))

count += map.get(target - nums[i]);

if (!map.containsKey(nums[i]))

map.put(nums[i], 1);

else map.put(nums[i], map.get(nums[i]) + 1);

}

return count;

}

public static void main(String[] args) {

int rvalue = TwoSumCount(new int[] {1, 1, 2, 3, 4}, 5);

System.out.println(rvalue);

return;

}

}

public int[] twoSum(int[] nums, int target) {

HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

for(int i = 0; i < nums.length; i++){

int key = target - nums[i];

if(hm.containsKey(key)){

return new int[]{hm.get(key) + 1, i + 1};

}

hm.put(nums[i], i);

}

return new int[]{0 , 0};

}