debug Q7

1. ArraySum: 计算一个整数数组的和

bug: sum = arr[i], should be sum += arr[i]

2. CountOccurence: 统计某数在整数数组中出现的次数。

bug: while loop 里加 i++

3. CheckGrade: 根据百分制分数计算GPA

bug: 判断条件 "||" should be "&&"

4. RemoveElement: 删除数组中特定位置元素，超出范围则返回原数组

bug: 第七行arr[i++] should be arr[i+1]

```
public static int[] removeElement(int arr[], int index) {
            int i, j, len = arr.length;
            if (index < len) {
                  for (i = index; i < len - 1; i++) {
                        arr[i] = arr[i+1];
                  }
                  int rarr[] = new int[len - 1];
                  for (i = 0; i < len - 1; i++) {
                         rarr[i] = arr[i];
                  }
                  return rarr;
            } else
                  return arr;
      }
```

5. DigitCount: 将一个整数除以它的位数取余数 num%len(num) == 0 or != 0

bug: 错在最后返回的除法时保留原数就行把num用另一个变量存起来就好因为num最后变成了0

```
public int find(int num) {
      int count = 0;
       while(num != 0) {
             num = num/10;
             count++;
        }
       return num%count;
}
```

6. Sort Array

```
public static int[] sortArray(int[] arr) {
            int len = arr.length;
            int small, pos, i, j, temp;
            for (i = 0; i <= len - 1; i++) {
                  for (j = i; j < len; j++) {
                        temp = 0;
                        if (arr[i] < arr[j]) {
                              temp = arr[i];
                              arr[i] = arr[j];
                              arr[j] = temp;
                        }
                  }
            }
            return arr;
      }
```

7. Sort Array 2

```
public static int[] sortArray2(int arr[]) {
		int i, max, location, j, temp, len = arr.length;
		for (i = 0; i < len; i++) {
			max = arr[i];
			location = i;
			for (j = i; j < len; j++) {
				if (max < arr[j]) {
					max = arr[j];
					location = j;
				}
			}
			temp = arr[i];
			arr[i] = arr[location];
			arr[location] = temp;
		}
		return arr;
	}
```
8. replace values 数组长度如果是偶数就把元素都改成1，奇数就都改成0

bug: for loop 里的 i <= len j <= len 把 <= 改成 <
```
public static int[] replaceValues(int arr[]) {
		int i, j, len = arr.length;
		if (len % 2 == 0) {
			for (i = 0; i < len; i++) {
				arr[i] = 0;
			}
		} else {
			for (j = 0; j < len; j++) {
				arr[j] = 1;
			}
		}
		return arr;
	}
```
9. reverse array

但class的名字居然是sort array

bug: **arr[len-1] 改成 arr[len-i-1] for循环里的len +=1去掉**
```
public static int[] reverseArray(int[] arr) {
		int i, temp, originalLen = arr.length;
		int len = originalLen;
		for (i = 0; i < originalLen / 2; i++) {
			temp = arr[len - i - 1];
			arr[len - i - 1] = arr[i];
			arr[i] = temp;
			// len +=1;
		}
		return arr;
	}
```
10. EvenOdd pattern

for循环缺大括号
```
public static void print4(int num) {
		int i, print = 0;
		if (num % 2 == 0) {
			print = 0;
```

```
                for (i = 0; i < num; i++) {
                     System.out.print(print + " ");
                     print += 2;
                }
           } else {
                print = 1;
                for (i = 0; i < num; i++) {
                     System.out.print(print + " ");
                     print += 2;
                }
           }
      }
```
11. Manchester: 输入一个01数组，输出manchester编码 假设第一个elem前的数字是0

bug:里面就一个判断语句，把判断语句的 `==` 改成 `!=` (考虑 `if (i == 0) ?`)
```
result = (A[i] == A[i - 1]); // bug
output[i] = result ? 1 : 0
```
12. Selection sort

bug: `if`里面的判断它写的是`arr[min]> arr[x]`,改成`arr[min]> arr[y]`
```
for (x = 0; x < n; x++) {
    int index_of_min = x;
    for (y = x; y < n; y++) {
        if (arr[index_of_min] > arr[x]) {
            y = index_of_min;
        }
        int temp = arr[x];
        arr[x] = arr[index_of_min];
        arr[index_of_min] = temp;
    }
}
// selection sort in ascending order 升序
for (int x = 0; x < n; x++) {
    int index_of_min = x;
    for (int y = x; y < n; y++) {
        if (arr[index_of_min] > arr[y]) {
            index_of_min = y;
         }
    }
    int temp = arr[x];
    arr[x] = arr[index_of_min];
    arr[index_of_min] = temp;
}
// bubble sort in descending order 降序
for (int x = 0; x < n; x++) {
    for (int y = x; y < n; y++) {
        if (arr[x] < arr[y]) {
            int temp = arr[x];
            arr[x] = arr[y];
            arr[y] = temp;
        }
    }
}
// ?
```

```
for (x = 0; x < n; x++) {
     int index_of_min = x;
     for (y = x; y < n; y++) {
         if (arr[index_of_min] <= arr[x]) {
              y = index_of_min;
              int temp = arr[x];
              arr[x] = arr[index_of_min];
              arr[index_of_min] = temp;
         }
     }
}
```
13. insertion sort < 改 > 或者 > 改 <
```
            for (int i = 1; i < n; i++) {
                  if (arr[i - 1] > arr[i]) {
                        int temp = arr[i];
                        int j = i;
                        while (j > 0 && arr[j - 1] > temp) {
                              arr[j] = arr[j - 1];
                              j--;
                        }
                        arr[j] = temp;
                  }
            }
```
14. PrintPattern2

a

ab

abc
```
public static void print2(int row) {
            for (int i = 1; i <= row; i++) {
                  char ch = 'a';
                  char print = ch;
                  for (int j = 0; j <= i; j++) {
                        System.out.print((print++));
                  }
                  System.out.println();
            }
      }
char ch = 'a';
char print = ch;
for(int j = 0; j <=i; j++) {
    System.out.print((ch++));
}
System.out.println("");
```
改为
```
char ch = 'a';
char print = ch;
for(int j = 0; j <=i; j++) {
    System.out.print((print++));
}
System.out.println(“”);
```

bug: 错在它把 `char ch = 'a'` 定义放在了 `for loop` 前，移进来就好 同时删掉了没用的变量
15. PrintPattern3
有一道题是要给一个row number，输出相应行数的pattern。比如：
```
row number = 3的话，就输出
11
1111
111111
row number = 2的话，就输出
11
1111
public static void print3(int row) {
            int x = 1;
            for (int i = 1; i <= row; i++) {
                  for (int j = i; j > 0; j--) {
                        System.out.print(x + "" + x);
                  }
                  System.out.println();
            }
      }
```
bug:第一层`for`循环少了大括号，导致最后那个`System.out.println()`最后只执行一次
16. remove duplicates form unsorted array 循环下标要从 `i + 1`开始
17. Array奇数偶数 `for loop i+=2` 改成 `i++`

reasoning Q24
KIDNEY:YENDIK::MUSCLE : (ELCSUM) 反一下
ASS ERT IVE NES S:SEN SSA EVI S TRE::MUL TIN ATI ONA L: (ANO LUM ITA L NIT) 三个字母一组
4 5 12 8 9 : 3 4 1 7 8::13 21 13 2 1 9 : 2 10 2 1 26 8 (BJBAZH) 每一位-1
VALIANT:UBKJZOS::TRANSCEND: (SSZORDDOC) +1-1
FU:DW:CX:(NR) 6 21 4 23 3 24 14 18 first 3 sum is 27
PRS, TVX, FIK, (LME) 排异 16 18 19 20 22 24 6 9 11 12 13 5 [A ceil((A+B)/2) B]
COMPUTER PMOCRETU TELEVISION (VELETNOISI)
(RHCAI) OEST HNDA ADEH
ADF (MPR) ILN EHJ 元音开头
STV XYA (KKT) BDE
0 2 3 5 8 10 15 17 24 26 (35)
27 24 64 (60) 3^3 - 3 = 24 4^3 - 4 = 60
4 12 6 18 12 36 30 (90) X 3
1 5 7 (8) 1+2^2 = 5 5+2^1 = 7 7+2^0 = **8**
1 1 4 2 13 3 40 4 (121) 40 + 3^4 = **121**
5 9 16 29 (54) 5*2-1 9*2-2 16*2-3 29*2-4
956:794:884:(678) 排异 前几个位数和20
10 74 202 394 (650) 间距递增 64
0 1 1 2 4 8 (16)
24576 6144 1536 384 (96) 间距 / 4
2 3 7 8 13 14 (20)

3 11 25 45 (71) 间距 8 14 20 26
0,1,1,2,3,7 (22) 0*1 + 1 = 1 1*1 + 1 = 2 1*2 + 1 = 3 2*3 + 1 = 7 3*7 + 1 = 22
16 30 46 62 (82) 13+3 29+1 43+3 61+1 79+3 or 16+30 16+46 16+62=78
1 4 27 256 (3125) n^n
ADP, QTX, HKR, (STE) ?
ADP QTS HKR STE (QTS or ADP完全平方数)
ADP = 1， 4， 16 = 00001， 00100， 10000， 位与的结果是0
QTS = 17，20，19 = 10001， 10100， 10011，位与的结果是10000
HKR = 8，11，18 = 01000，01011，10010，位与的结果是0
STE = 19，20，5 = 10011，10100，00101，位与的结果是0

方向题 take an about-turn, turn around, 意思是reverse turn.

Candidate/录取新员工
An IT company has decided to recruit software developers. Conditions for selection of a candidate are as follows:
The candidate should have at least a bachelor’s degree in engineering.
The candidate should have scored at least 60% marks in his/her bachelor’s degree and 80% marks in 12th grade.
The candidate must have at least 1 year’s work experience.
The candidate should be writing to sign a bond of 2 years.
The candidate should not be more than 28 years and not less than 21 years of age as on 01.02.2012

八产品 一个公司有4个子公司，每个子公司有2个产品 问关系
A manufacture company has 8 products and 4 divisions. Four divisions are lead by Alan, Betty, Cathy, Diana. The 8 products are: mixer, iron, water pump, geyser, juicer, blender, grinder, and heater. Each division produces 2 products, no 2 divisions produces the same product. **Diana**’s division produced **Geyser**, **Cathy**’s division produces **water pump**. **Mixer** and **iron** are produced by division lead by **Alan** and **Betty** respectively 分别的. The division that produces **mixer** doesn’t produce **blender**.
Four questions:
Q1 if the division that produces **mixer** doesn’t produce **juicer**, which of the following statement is true? if A - mixer (grinder or heater) B iron if B - mixer A iron
Q2 如果alan生产mixer和heater,那么betty生产什么 B iron (juicer blender grinder)
Q3 if the division that produces **mixer** also produces **juicer**,how many ways are there for product pairs? (3! = 6) C water D geyser B iron - (blender, grinder, heater)

八仙桌
A, B, C, D, E, F, G H are sitting around a round table. ‘F’ is two places to the right of ‘C’, ‘A’ and ‘E’ are on either side of ‘G’, ‘B’ and ‘H’ are opposite to each other. ‘C’ is facing north.
anti clock direction possible arrangement？ AHCDFBEG
what is opposite to D ? G
哪两个坐对面？ D & G
谁坐D旁边？ C

6人团团坐 SGRMNA这几个围着坐，GAS不能相互对着坐 两种 GAS间隔一人坐 or 全挨着坐
if R 在AS中间 R对面 G
if G 左右是AR 问A对面 M or N

M1,M2,M3,M4,M5和W1,W2,W3。出差必须派至少三男一女。M1和M3不能共存，M4和W2不能共存 4个题
Q1 问如果派了M2和M3和W2，还可以派谁 M5 W3
Q2 问如果M1 M2去了，还可以派谁 M4 M5 W1 W3 or M5 W1 W2 W3
Q3 如果去了四个男生，那么谁不能选 W2

四个人L, C, M, N 在一个长方形房间的四个角上。 coffee vending machine is situated at one of the corners and a restroom at another corner of the meeting hall. L and C are at either sides of the white board, which is situated at the centre of the side which is opposite to the side at whose corners the coffee vending machine and the restroom are located. Coordinator M is not at the corner where the restroom is located.
谁跟谁对角是可能的？
即平面图是这样的

```
L/C------WHITE BOARD-------C/L
I                           I
I                           I
I                           I
M--------------------------------N
COFFEE                      RESTROOM
VENDER
```

Q1
1. Lily is not on the side of the hall where the white board is placed
是说L不在WHITE BOARD放置的那一侧，错
2. Nina is adjacent to the restroom at one corner
是说N与RESTROOM相邻，在某一CORNER，对
3. Cathy is at the corner, adjacent to the coffee vending machine
是说C在某一个CORNER，那个CORNER与COFFEE VENDING MACHINE 相邻，所以不是COFFEE VENDING MACHINE的那个CORNER，对
4. Mary is adjacent to the coffee vending machine, at one corner of the hall
是说M与COFFEE MACHINE 相邻，在某一个CORNER，对
5. Lily is at the corner, adjacent to the coffee machine
是说L在某一个CORNER，那个CORNER与COFFEE MACHINE相邻，对
Q2 which of the following pairs are at the diagonal corners
Restroom and Nina
Nina and Lily
Mary and coffee vending machine
Cathy and Lily
Nina and Mary

Conditions for appointing a distributor, for petroleum gas 石油气 throughout **Georgia**
1 要是American

2. 21-50岁 (2008.9.5,按这个年份计算)
3. 学历至少要high school
4. 要在Georgia住5年以上 immediately preceding the date of application
5. 家庭年收入 not more than $30,000
6. 自己不是oil company 的dealer
7. 也没有亲属是oil company的dealer.
8. restrictions related to **annual income** would **not be applicable** to applications working in corporations, owned or controlled by state department. Such a case should be referred to the Managing Director
9. for unemployed applications who hold at least a bachelor's degree, conditions 6 and 7 may be waived
10. if an applicant is from a rural district 乡下 but is not a resident of Georgia, the case may be referred to the Chairman
Q1 WT who works in a public corporation owned by a state department, is an American. 23 years old, hold bachelor's degree and has an annual income of $35,000. he has been staying in Georgia for 7 years. Neither he nor any of his relatives works as a distributor or a dealer for any oil company. 交给manager把
Q2 Anna, a non-Georgia, American citizen, is a high school graduate with family income of $20,000 per annum 年, her date of birth is 15.03.1984. she does not have dealership of any oil company nor doest she have any close relative as a dealer or a distributor. she lives in a rural district. 交给chairman把

分房问题
1. 至少10年在公司工作工作,其中至少4年是manager
2. 包括自己在内,family人数不能超过5人
3. 以58岁为退休年龄,至少还要工作5年才能退休
4. 没有自己的house
5. 不能住在配偶 mate spouse 的house里
6. 如果仅不满足1,而且还是manager的话,上报给Finance Director
7. 如果仅不满足3,而且还是senior manager的话,上报给Manager Director
8. 如果是从外地搬来的,可以不满足条件1.
Q1 Jonh家里3个人,没有房子,作为senior manager 4年,一共工作12 年,2020年退休 (题目是出自 2014 7 30),问是否满足
Q2 老王家里3个人租房子住,作为manager 5年,工作11年,年龄42岁
Q3 第二问那个女的没有说家里有几个人，所以我选了信息不足
关键在于children，这是复数，可能有2个孩子，也可能有4个孩子

招PM
1. 本科是学CS的
2. 有MBA学位
3. 本科GPA 3.0+
4. 如果没有MBA学位,但是工作5年以上,需上报HR

5. 本科不是学CS,但是在CS相关工作3年以上,上报HR
那么,请问:闰土本科学热水锅炉维修的,GPA 4.0,没有念过MBA,在Google修了5年的锅炉,当 了3年的程序员,则应该: 上报HR

招聘问题，要求学士学位并且B以上，MBA并且B以上，12年级成绩70%以上，一年工作经验，30岁以下，这题也有两问，第二问那人说有学士学位但是没说成绩，所以我也选了信息不足
推荐入职（有一差点没反应过来，第二个case学位其他都符合但没给具体grade）

Buyme和INXsell两个卖东西的网站
区别是Buyme允许买卖二手。人们可以在上面卖book, bike, treadmill. 卖家设最低价,买家竞价,最后标得最高价的人必须买下,否则可以采取司法程序处理。 但Buyme对于拍卖的物品没有保障。INXsell可以直接从生产商购买产品
A. 两个网站都是online buying and selling 的web-portals
B. buyme上仅仅可以拍卖book, bike, treadmill这几样二手物品。
C. 两个网站唯一区别是,Buyme允许买二手物品
D. buyme上出售的物品lowest price可以低至0
E. 竞得最高价的人必须要买下商品
选的是最后一个拍下的人必须买下

Saira一直都是受同事们敬仰的一位好员工,直到她因为bipolar综合征带来 的问题需要休假一段时间。不久后她就被炒鱿鱼了,公司给出的理由是她的 病情导致她工作效率低下并且会无意识地冒犯上级(bipolar综合征的症状之 一就是在无意识的情况下说各种话说个不停)。而最终法院裁定,Saira被炒 鱿鱼是因为公司相信了关于这种病的种种myth(就是歧视)。问哪个选项是对的?(**选Saira被炒鱿鱼不是因为她自己有错**,其余几个选项都是空穴来风)

太阳能公司
1 have Environment clearance certificate ECC
2 developed at least 3 solar products
3 none of its products made from synthetic 合成polymers
4 headquarters in Texas
5 have a grade A certified unit of its products
6 not have any legal dispute 阻止 related to land or forest, pending against them
Q1 US based company, sell batteries, has ECC, currently produce two solar products, both grade A. not use synthetic polymers, no legal case pending 不邀请 (产品<3)
Q2 headquarter in Texas, sell solar products including bulbs, cookers, and heaters, not made of synthetic polymers, has ECC, products are grade A, not have any legal case pending. 邀请

想要获得promotion，给出以下10个条件：
1. 至少工作5年
2. 至少获得学士学位
3. 写作成绩不低于60
4. Group discussion成绩不低于65

5. Interview成绩不低于 60
6. 无犯罪记录
7. 年龄在30～45岁之间
8. 学术大于60
9. 如果不满足3，而且group discussion成绩不低于75，interview不低于70，上报给general manager
10. 如果不满足8，上报给executive director

一楼有3层，一共66人，第二层最多，1 其中一层21人 2 第二层比第一层多2人 21 22 (23)
印度radio那个，看都没看就选了B saturation radio 指同一个广告反复宣传
delivery charge for goods bought from ABC company $30 avail 效益 bulk 大量的
卖冰箱 refrigerator
what date was the car purchased by Lily 1016 - 1019 1017 - 1020 两个条件加一块可以
4种元素 半径 radius
editor writer
定义新运算（V＃X）/（V＋X） only 1 X＜Z is true
A%B C=E D*B 1 A#D 2 C*E both false
a better sales person would be the one who is able to explain the features of his/her product in a simple manner
真人秀 (答案 参加的人通过做bizarre滑稽 的事情得到钱）
leap year 29 common year 28

```
coding
import java.util.*;
```

overlap rectangle

```
public class OverLap {
	// rectangle A, B
	// time O(1), space O(1)
	public static boolean check(Node topLeftA, Node topLeftB, Node
bottomRightA, Node bottomRightB) {
		// if one rectangle is on left side of other
		if (bottomRightA.y <= topLeftB.y || bottomRightB.y <=
topLeftA.y) {
			return false;
		}
		// if one rectangle is above other
		if (topLeftA.x >= bottomRightB.x || topLeftB.x >=
bottomRightA.x) {
			return false;
		}

		return true;
	}

	public static class Node {
		double x;
		double y;
		public Node(double x, double y) {
			this.x = x;
			this.y = y;
		}
	}
}

// return !(Bx <= Cx || Ay <= Dy || Ax >= Dx || By >= Cy);
```

K closest points

```
import java.util.Comparator;
import java.util.PriorityQueue;
import java.util.HashMap;
import java.util.PriorityQueue;
	 // SOL 1
	public static CPoint[] findClose1(CPoint[] arr, int k) {
		if (k == 0) {
			return new CPoint[0];
		}
		// construct a min heap to store and sort incoming dis
	  // initial size also can be arr.length
		PriorityQueue<CPoint> minHeap = new
PriorityQueue<CPoint>(k, new Comparator<CPoint>() {
			@Override
			public int compare(CPoint a, CPoint b) {
				double disa = a.x * a.x + a.y * a.y;
				double disb = b.x * b.x + b.y * b.y;
				if (disb > disa) {
```

```
                    return -1;
                } else if (disb < disa) {
                    return 1;
                } else {
                    return 0;
                }
            }
        });

        for (CPoint p : arr) {
            minHeap.offer(p);
        }
        // return the k closest points
        CPoint[] rst = new CPoint[k];
        for (int i = 0; i < k; i++) {
            rst[i] = minHeap.poll();
        }
        return rst;
    }
    // SOL 1'
    public static CPoint[] findClose3(CPoint[] arr, int k) {
        if (k == 0) {
            return new CPoint[0];
        }

        Comparator<CPoint> cmp = new Comparator<CPoint>() {
            public int compare(CPoint o1, CPoint o2) {
                double d1 = o1.x * o1.x + o1.y * o1.y;
                double d2 = o2.x * o2.x + o2.y * o2.y;
                if (d1 > d2) {
                    return 1;
                } else if (d1 == d2) {
                    return 0;
                } else {
                    return -1;
                }
            }
        };

        PriorityQueue<CPoint> minHeap = new PriorityQueue(k, cmp);
        for (CPoint p : arr) {
            minHeap.offer(p);
        }

        CPoint[] rst = new CPoint[k];
        for (int i = 0; i < k; i++) {
            rst[i] = minHeap.poll();
        }
        return rst;
    }
    // SOL 2
    public static CPoint[] findClose2(CPoint[] arr, int k) {
        if (k == 0) {
            return new CPoint[0];
        }
        CPoint[] rst = new CPoint[k];
        DPoint[] sort = new DPoint[arr.length];
        for (int i = 0; i < arr.length; i++) {
            sort[i] = new DPoint(arr[i]);
        }

        Arrays.sort(sort);

        for (int i = 0; i < k; i++) {
            rst[i] = sort[i].node;
        }
        return rst;
    }

    public static class DPoint implements Comparable<DPoint> {
        double distance;
        CPoint node;
        DPoint(CPoint m) {
            distance = m.x * m.x + m.y * m.y;
            node = m;
        }

        @Override
        public int compareTo(DPoint o) {
            if (this.distance > o.distance) {
                return 1;
            } else if (this.distance < o.distance) {
                return -1;
            } else {
                return 0;
            }
        }
    }
    // SOL 3
    public static CPoint[] findClose3(CPoint[] arr, int k) {
        if (arr == null || arr.length == 0 || k == 0) {
            return new CPoint[0];
        }

        double[] dis = new double[arr.length];
        for (int i = 0; i < arr.length; i++) {
            dis[i] = arr[i].x * arr[i].x + arr[i].y * arr[i].y;
        }

        HashMap<Double, CPoint> map = new HashMap<Double,
CPoint>();
        for (int i = 0; i < arr.length; i++) {
            map.put(dis[i], arr[i]);
        }
        CPoint[] rst = new CPoint[k];
        Arrays.sort(dis);
        for (int i = 0; i < k; i++) {
            if (map.containsKey(dis[i])) {
                rst[i] = map.get(dis[i]);
            }
        }
        return rst;
    }

    public static class CPoint {
        double x;
        double y;
        public CPoint(double a, double b) {
            x = a;
            y = b;
        }
    }
```

window sum

```
import java.util.ArrayList;
import java.util.List;
        // SOL 1 sliding window
        // runtime is O(nk)
        public static List<Integer> getSum1(List<Integer> arr, int k) {
            List<Integer> rst = new ArrayList<Integer>();
            if (arr == null || arr.size() == 0 || k <= 0) {
                return rst;
            }
            // use two pointers to maintain the window size
            int left = 0;
            int right = left + k - 1;
            // sum up the values within the window
            while (right < arr.size()) {
                int sum = 0;
                for (int i = right; i >= left; i--) {
                    sum += arr.get(i);
                }
                rst.add(sum);
                left++;
                right++;
            }
            return rst;
        }

public static List<Integer> getSum3(List<Integer> arr, int k) {
            List<Integer> rst = new ArrayList<Integer>();
            // k > arr.size() ?
            if (arr == null || arr.size() == 0 || k == 0) {
                return rst;
            }
            int sum = 0;
            for (int i = 0; i < k; i++) {
                sum = sum + arr.get(i);
            }

            rst.add(sum);
            for (int i = k; i <= arr.size() - 1; i++) {
                  sum = sum + arr.get(i) - arr.get(i - k);
                  rst.add(sum);
            }
            return rst;
      }

      // SOL 2
      // runtime is O(nk), n is the size of arr, space used is O(1)
      public static List<Integer> getSum2(List<Integer> arr, int k) {
            List<Integer> rst = new ArrayList<Integer>();
            // consider the corner case
            if (arr == null || arr.size() == 0 || k <= 0) {
                  return rst;
            }
            // use a variable to track the tail index when moving
window
            int cnt = 0;
            for (int i = 0; i < arr.size(); i++) {
                  cnt++;
                  if (cnt >= k) {
                        int sum = 0;
                        // Once finish sum up the values within window
                        for (int j = i; j >= i - k + 1; j--) {
                              sum += arr.get(j);
                        }
                        rst.add(sum);
                  }
            }
            return rst;
      }
```

longest palindromic substring

状态表达式：`D[i][j]` 表示`i,j`这2个索引之间的字符串是不是回文.

```
public class LongestSub {
      // SOL 1
      // runtime is O(n^2), space is O(n^2)
      public static String longestSubstr1(String s) {
            if (s == null || s.length() == 0) {
                  return null;
            }
            String rst = null;
            int max = 0;
            // D[i][j] represents that whether substring palindromic
between i j
            boolean[][] D = new boolean[s.length()][s.length()];
            for (int j = 0; j < s.length(); j++) {
                  for (int i = 0; i <= j; i++) {
                        D[i][j] = s.charAt(i) == s.charAt(j) && (j - i <=
2 || D[i + 1][j - 1]);
                        if (D[i][j]) {
                              if (j - i + 1 > max) {
                                max = j - i + 1;
                                rst = s.substring(i, j + 1);
                            }
                        }
                    }
                }
                return rst;
            }
            // SOL 2
            public static String longestSubstr2(String s) {
                if (s == null || s.length() == 0) {
                    return s;
                }

                int len = s.length();
                int max = 0;
                String rst = "";
                // expand string to 2 times length
              // assume there is one more char '#' adjacent to every char in
        the string
                for (int i = 1; i <= 2 * len - 1; i++) {
                    int cnt = 1;
                    // consider every char in the string as center and
        check whether its neighbors equal
                 // palindrome check
                    while (i - cnt >= 0 && i + cnt <= 2 * len && get(s, i
        - cnt) == get(s, i + cnt)) {
                        cnt++;
                    }
                    // there will be one extra cnt for the outbound
                    cnt--;
                    if (cnt > max) {
                        rst = s.substring((i - cnt) / 2, (i + cnt) / 2);
                        max = cnt;
                    }
                }
                return rst;
            }
            // SOL 3 Manchester O(n) runtime
            public static String longestSubstr3(String s) {
                String str = preProcess(s);
                int len = str.length();
                int[] pos = new int[len];

                int center = 0;
                int right = 0;
                for (int i = 1; i < len - 1; i++) {
                    // j and i are symmetric around center
                    int j = 2 * center - i;
                    if (right > i) {
                        pos[i] = Math.min(right - i, pos[j]);
                    } else {
                        pos[i] = 0;
                    }
                    // expand palindrome centered at i
                    while (str.charAt(i + 1 + pos[i]) == str.charAt(i - 1
        - pos[i])) {
                        pos[i]++;
                    }
                    // if palindrome centered at i expand past right
                    // then adjust center based on expand palindrome
                    if (i + pos[i] > right) {
                        center = i;
                        right = i + pos[i];
                    }
                }
                // find the longest palindrome
                int maxLen = 0;
                int centerIndex = 0;
                for (int i = 1; i < len - 1; i++) {
                    if (pos[i] > maxLen) {
                        maxLen = pos[i];
                        centerIndex = i;
                    }
                }
                centerIndex = (centerIndex - 1 - maxLen) / 2;
                return s.substring(centerIndex, centerIndex + maxLen);
            }
            // s = "abcdefg", then the tag = "^#a#b#c#d#e#f#g#$"
            private static String preProcess(String s) {
                if (s == null || s.length() == 0) {
                    return "^$";
                }
                StringBuilder tag = new StringBuilder("^");
                for (int i = 0; i < s.length(); i++) {
                    tag.append("#").append(s.substring(i, i + 1));
                }
                tag.append("#$");
                return tag.toString();
            }

            private static char get(String s, int i) {
                if (i % 2 == 0) {
                    return '#';
                } else {
                    return s.charAt(i / 2);
                }
            }

            public static void main(String[] args) {
                System.out.print(longestSubstr3("xyyxabcdcba"));
            }
        }
```

merge two linked list

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        ListNode dummy = new ListNode(0);
        ListNode cur = dummy;
        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;
        }
        if (l1 != null) {
            cur.next = l1;
        } else {
            cur.next = l2;
        }
        return dummy.next;
    }
}
```

two sum count

```
Consider case: {3, 2, 4} target = 6, output {3, 3} {2, 4}
import java.util.HashMap;
    // SOL 1
    // time O(n), space O(n)
     public static int count1(int[] nums, int target) {
           // corner case
           if (nums == null || nums.length < 2) {
                 return 0;
           }

           int cnt = 0;
           HashSet<Integer> hash = new HashSet<Integer>();
           for (int i = 0; i < nums.length; i++) {
                 hash.add(nums[i]);
           }

           for (int i = 0; i < nums.length; i++) {
                 if (hash.contains(target - nums[i])) {
                       cnt++;
                 }
           }
           return cnt;
     }
```

```
      // SOL 2
      // pairs 不算重复的 数值
      public static int count2(int[] nums, int target) {
            if (nums == null || nums.length < 2) {
                  return 0;
            }

            HashMap<Integer, Integer> map = new HashMap<Integer, 
Integer>();
            for (int i = 0; i < nums.length; i++) {
                  map.put(nums[i], i);
            }

            int cnt = 0;
            for (int i = 0; i < nums.length; i++) {
                  if (map.containsKey(target - nums[i])) {
                        if (i != map.get(target - nums[i])) {
                              cnt++;
                              map.remove(nums[i]);
                              map.remove(target - nums[i]);
                        }
                  }
            }
            return cnt;
      }
      // SOL 3
      // 算重复的
      public static int count3(int[] nums, int target) {
            if (nums == null || nums.length < 2) {
                  return 0;
            }

            HashMap<Integer, Integer> map = new HashMap<Integer, 
Integer>();
            int cnt = 0;
            for (int i = 0; i < nums.length; i++) {
                  if (map.containsKey(target - nums[i])) {
                        cnt += 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 cnt;
      }
```

Check Subtree
Determine whether T2 is subtree of T1

```
// SOL 1
// time O(nm) or O(n + km) k is the number of occurrences of T2's root 
in T1
```

```
// space O(log n + log m) recursion
public static int check1(TreeNode t1, TreeNode t2) {
        if (t2 == null) {
            return 1;
        }
        if (subTree(t1, t2)) {
             return 1;
        } else return -1;
    }

    public static boolean subTree(TreeNode r1, TreeNode r2) {
        if (r1 == null) {
            return false;
        } else if (r1.val == r2.val && matchTree(r1, r2)) {
            return true;
        }
        return (subTree(r1.left, r2) || subTree(r1.right, r2));
    }

    public static boolean matchTree(TreeNode r1, TreeNode r2) {
        if (r2 == null && r1 == null) {
            return true;
        } else if (r1 == null || r2 == null) {
            return false;
        } else if (r1.val != r2.val) {
            return false;
        } else {
            return (matchTree(r1.left, r2.left) &&
matchTree(r1.right, r2.right));
        }
    }

    public static class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;
        public TreeNode(int x) {
            val = x;
        }
    }
```

Valid Parentheses

判断是不是valid的parenthesis string，不是的话返回-1，是的话返回valid pair个数

```
import java.util.Stack;
// time O(n), space O(n / 2)

public class Solution {
    public boolean isValid(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.isEmpty()) {
                        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.isEmpty();
      }
}

public class Solution {
    public boolean isValid(String s) {
        if (s == null || s.length() == 0) {
            return true;
        }

        Stack<Character> stack = new Stack<Character>();
        int len = s.length();
        for (int i = 0; i < len; i++) {
            char c = s.charAt(i);
            if (stack.isEmpty()) {
                if (c == ')' || c == ']' || c == '}') {
                    return false;
                }
                stack.push(c);
                continue;
            }

            if (c == ')' && stack.peek() == '('
            || c == ']' && stack.peek() == '['
            || c == '}' && stack.peek() == '{') {
                stack.pop();
            } else if (c == '(' || c == '[' || c == '{') {
                stack.push(c);
            } else {
                return false;
            }
        }
        return stack.isEmpty();
    }
}

public class Solution {
    public boolean isValid(String s) {
        if (s == null || s.length() % 2 == 1) {
            return false;
        }

        HashMap<Character, Character> map = new HashMap<Character,
Character>();
```

```
        map.put('(', ')');
        map.put('[', ']');
        map.put('{', '}');
        /*
        map.put('a', '1');
        map.put('b', '2');
        */
        Stack<Character> stack = new Stack<Character>();
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (map.containsKey(c)) {
                stack.push(c);
            } else if (map.containsValue(c)) {
                if (!stack.isEmpty() && map.get(stack.peek()) == c) {
                    stack.pop();
                } else {
                    return false;
                }
            }
        }
        return stack.isEmpty();
    };
}
```

Reverse second half of linked list
e.g. 1-2-3-4-5-6 change to 1-2-3-6-5-4, 1-2-3-4-5 change to 1-2-5-4-3

```
      // time O(n), space O(1)
      public static JNode reorder1(JNode head) {
           if (head == null || head.next == null || head.next.next ==
null) {
                return head;
           }

           JNode fast = head.next;
           JNode slow = head;
           while (fast.next != null && fast.next.next != null) {
                fast = fast.next.next;
                slow = slow.next;
           }
           JNode pre = slow.next;
           JNode cur = pre.next;
           while (cur != null) {
                pre.next = cur.next;
                cur.next = slow.next;
                slow.next = cur;
                cur = pre.next;
           }
           return head;
      }

     public static JNode reorder2(JNode head) {
           if (head == null || head.next == null) {
                return head;
```

```
            }
            JNode pre = findMidPre(head);
            JNode right = pre.next;
            pre.next = null;
            right = reverse(right);
            pre.next = right;
            return head;
      }
     public static JNode reverse(JNode node) {
            if (node == null) {
                  return node;
            }
            JNode dummy = new JNode(0);
            while (node != null) {
                  JNode tmp = node.next;
                  node.next = dummy.next;
                  dummy.next = node;
                  node = tmp;
            }
            return dummy.next;
      }
      public static JNode findMidPre(JNode node) {
            if (node == null) {
                  return node;
            }
            JNode fast = node.next;
            JNode slow = node;
            while (fast != null && fast.next != null &&
fast.next.next != null) {
                  fast = fast.next.next;
                  slow = slow.next;
            }
            return slow;
      }

      public static class JNode {
            int val;
            JNode next;
            JNode(int x) {
                  val = x;
            }
      }
```

gray code

```
public class GrayCode {
      // if can be placed successively in a gray code sequence return 1
else 0
      public static int grayCheck(byte term1, byte term2) {
            // XOR operation by bit
            // for gray code there is only one diff between two terms
            byte rst = (byte)(term1 ^ term2);
            for (int i = 0; i <= 7; i++) {
                  byte tmp = (byte)(1 << i);
```

```
                if (tmp == rst) {
                     return 1;
                }
           }
           return 0;
      }

      public static void main(String[] args) {
           byte term1 = (byte)0x0a, term2 = (byte)0x80;
           int out;
           out = grayCheck(term1, term2);
           System.out.println(out);
      }
}
```

rotate string

```
//SOL 1
public static int isRotation2(String word1, String word2) {
           if (word1 == null || word2 == null || word1.length() == 0
|| word2.length() == 0 || word1.length() != word2.length()) {
                return -1;
           }

           String str = word1+word2;
           return str.contains(word2) ? 1 : -1;
           // return str.indexOf(word2) ? 1 : -1;
      }
//SOL 2
public static boolean isRotation1(String s1, String s2) {
           int len = s1.length();

           if (len == s2.length() && len > 0) {
                String s1s1 = s1 + s1;
                return isSubstring(s1s1, s2);
           }
           return false;
      }

      private static boolean isSubstring(String big, String small) {
           if (big.indexOf(small) >= 0) {
                return true;
           } else {
                return false;
           }
      }
```

remove vowel

```
// indexOf Returns the index within this string of the first
occurrence of the specified character or -1 if the character does not
occur
import java.util.Arrays;
import java.util.HashSet;
     public static String remove1(String s) {
           StringBuilder sb = new StringBuilder();
           HashSet<Character> hash = new
HashSet<Character>(Arrays.asList(
                new Character[]{'a', 'e', 'i', 'o', 'u', 'A', 'E',
'I', 'O', 'U'}));
           for (Character c : s.toCharArray()) {
                 if (hash.contains(c)) {
                      continue;
                 }
                 sb.append(Character.toString(c));
           }
           return sb.toString();
     }

     public static String remove2(String s) {
           StringBuilder sb = new StringBuilder();
           String vowel = "aeiouAEIOU";
           for (int i = 0; i < s.length(); i++) {
                 char c = s.charAt(i);
                 if (vowel.indexOf(c) == -1) {
                      sb.append(c);
                 }
           }
           return sb.toString();
     }
```

必须全对 排除法 代码一定要最优解 OA1OA2 新老题库都要做一遍 准备好电子字典 字母表
a code debugging test (20 min), a reasoning test (35 min) and a compiler based coding test (30 min)
if you need to take a break, the best time would be between the code debugging and reasoning test. If you choose to take a break, log out, and when you're ready to start, log back in with the credentials below and the test will take you where you left off last.
ONLINE ASSESSMENT INSTRUCTIONS
(1) Set aside 1.5 hrs. in a quiet location where you will not be interrupted
(2) Ensure you have a reliable internet connection
(3) Be prepared to have an ID verification photo taken via your webcam
(4) Use the latest version of one of Google Chrome (35) or Firefox (25+)
(5) Ensure your web browser allows pop-ups
(6) Be sure to test your webcam to ensure it is working properly
(7) When you're ready to take the test log in by typing in (NOT copying and pasting) credentials below
Username: yimingpeng@engin.umass.edu
Password: pbrntnu7

JDK 1.5 PriorityQueue