**Introduction to Computer Science**

**Homework 9**

**沙行勉教授**

**Due Date: 星期三，Dec. 12, 2018**

**Do not copy others, or you will get punished. If you allow other students to copy your report, you will be punished as well. In the homework, you will get more experience in programming. Good for you. 但是，对于编程题，我希望有基础的人能多帮助其他同学，帮助他们debug, 但是不要让他们抄袭。好吗？**

**PUT DOWN YOUR NAME and EMAIL ADDRESS IN YOUR REPORT.** tThe Filename of your report should be HOMEWORK N\_your Student ID\_your name \_Teacher’s last name (Sha or Zhang), where N is the homework number, for example, *1\_2018XXXX \_XiaoMing\_Sha*. Then send your report to the TA of your class. 注意，两班的TA不一样，千万不要送错了。**你可以用中文来回答问题。**

**请同学阅读书本的第五章部分。**

1. **（每题8 points）练习题5.1.1， 5.1.2， 5.1.3， 5.1.4， 5.1.5， 5.1.6**

**5.1.1：**

**#生成所需要的列表**

**import random**

**n = int(input("Enter the number of coins >=2:"))**

**w\_normal = random.randint(2,5)**

**index\_faked = random.randint(0,n-1)**

**L = [w\_normal]\*n**

**L[index\_faked] = w\_normal - 1**

**#开始寻找假币的索引**

**def findcoin\_2(L):**

**if len(L) <= 1:**

**print("Error:coins are too few")**

**return**

**i = 0**

**while i < len(L)-1 :**

**if L[i] < L[i+1]:**

**return i**

**elif L[i] > L[i+1]:**

**return i+1**

**i+=2**

**print("All coins are the same")**

**return -1**

**print(L)**

**print("The index of faked coin:", findcoin\_2(L))**

**5.1.2：**

**#生成所需要的列表**

**import random**

**n = int(input("Enter the number of coins >=2:"))**

**w\_normal = random.randint(2,5)**

**index\_faked = random.randint(0,n-1)**

**L = [w\_normal]\*n**

**L[index\_faked] = w\_normal - 1**

**#开始寻找假币的索引**

**def findcoin\_3(L):**

**L = L[:];i = 0**

**if len(L) < 1:**

**print("Error : coins are too few")**

**while len(L)>1:**

**if len(L)%2 == 0:**

**if sum(L[0:len(L)//2]) > sum (L[len(L)//2:]):**

**i+=len(L)//2;L = L[len(L)//2:]**

**elif sum(L[0:len(L)//2]) < sum (L[len(L)//2:]):**

**L = L[0:len(L)//2]**

**else:**

**return (-1)**

**else:**

**if sum(L[0:len(L)//2]) > sum (L[len(L)//2:len(L)-1]):**

**i+=len(L)//2;L = L[len(L)//2:len(L)-1]**

**elif sum(L[0:len(L)//2]) < sum (L[len(L)//2:len(L)-1]):**

**L = L[0:len(L)//2]**

**else:**

**if L[0] > L[-1]:**

**return i+len(L)-1**

**else:**

**return -1**

**return i**

**print(L)**

**print("The index of faked coin:", findcoin\_3(L))**

**5.1.3：**

**a是用来记录假币的索引的；**

**当列表长度为偶数且前半段总和等于后半段总和时，return -1；**

**当列表长度为奇数，且除去最后一个元素后前半段总和等于后半段总和，且最后一个元素还大于等于前面任意一个元素时，return -1**

**5.1.4**

**可以使用三分法，不断3分直至列表长度小于等于3，确定含假币的子列表，然后比较第一个而第二个元素的大小来判断是否有假币。**

**也可以使用k分法，优势在于可以迅速减小含假币的范围，劣势是所确定的子列表所含元素太多，仍要进行复杂的演算来确定具体位置**

**5.1.5**

**将原来的sum改成求平均值，即sum(L)/len(L),然后采用二分法，如果前半段的average等于后半段的，则每一段各含一个假币，然后就变成了求一个假币的情况，算法便可沿用上面的，否则，则average小的含两个假币，则再进行二分，直到分为各含1个或列表长度很小便停止**

**5.1.6**

**将列表进行3分，计算3堆金币的average，则必有两堆相等，另一堆如果比这两堆重，则将上面算法的大于小于号取反，反之则可直接沿用上述算法**

1. **（10 points）练习题 5.1.7**

**5.1.7**

**#生成所需要的列表**

**import random**

**n = int(input("Enter the number of coins >=2:"))**

**w\_normal = random.randint(2,5)**

**index\_faked = random.randint(0,n-1)**

**L = [w\_normal]\*n**

**order = input("请输入light or heavy:")**

**if order == "light":**

**L[index\_faked] = w\_normal - 1**

**if order == "heavy":**

**L[index\_faked] = w\_normal + 1**

**#针对假币比真币轻的算法**

**def findcoin\_small(L):**

**L = L[:];i = 0**

**if len(L) < 1:**

**print("Error : coins are too few")**

**while len(L)>1:**

**if len(L)%2 == 0:**

**if sum(L[0:len(L)//2]) > sum (L[len(L)//2:]):**

**i+=len(L)//2;L = L[len(L)//2:]**

**elif sum(L[0:len(L)//2]) < sum (L[len(L)//2:]):**

**L = L[0:len(L)//2]**

**else:**

**return (-1)**

**else:**

**if sum(L[0:len(L)//2]) > sum (L[len(L)//2:len(L)-1]):**

**i+=len(L)//2;L = L[len(L)//2:len(L)-1]**

**elif sum(L[0:len(L)//2]) < sum (L[len(L)//2:len(L)-1]):**

**L = L[0:len(L)//2]**

**else:**

**if L[0] > L[-1]:**

**return i+len(L)-1**

**else:**

**return -1**

**return i**

**#针对假币比真币重的算法**

**def findcoin\_big(L):**

**L = L[:];i = 0**

**if len(L) < 1:**

**print("Error : coins are too few")**

**while len(L)>1:**

**if len(L)%2 == 0:**

**if sum(L[0:len(L)//2]) < sum (L[len(L)//2:]):**

**i+=len(L)//2;L = L[len(L)//2:]**

**elif sum(L[0:len(L)//2]) > sum (L[len(L)//2:]):**

**L = L[0:len(L)//2]**

**else:**

**return (-1)**

**else:**

**if sum(L[0:len(L)//2]) < sum (L[len(L)//2:len(L)-1]):**

**i+=len(L)//2;L = L[len(L)//2:len(L)-1]**

**elif sum(L[0:len(L)//2]) > sum (L[len(L)//2:len(L)-1]):**

**L = L[0:len(L)//2]**

**else:**

**if L[0] < L[-1]:**

**return i+len(L)-1**

**else:**

**return -1**

**return i**

**print(L)**

**L1 = L[:len(L)//3];L2 = L[len(L)//3:2\*len(L)//3];L3 = L[2\*len(L)//3:]**

**n1 = sum(L1)/len(L1);n2 = sum(L2)/len(L2);n3 = sum(L3)/len(L3)**

**if n1 == n2:**

**if n3 < n1:**

**print("The index of faked coin:", findcoin\_small(L))**

**else:**

**print("The index of faked coin:", findcoin\_big(L))**

**elif n1 == n3:**

**if n2 < n1:**

**print("The index of faked coin:", findcoin\_small(L))**

**else:**

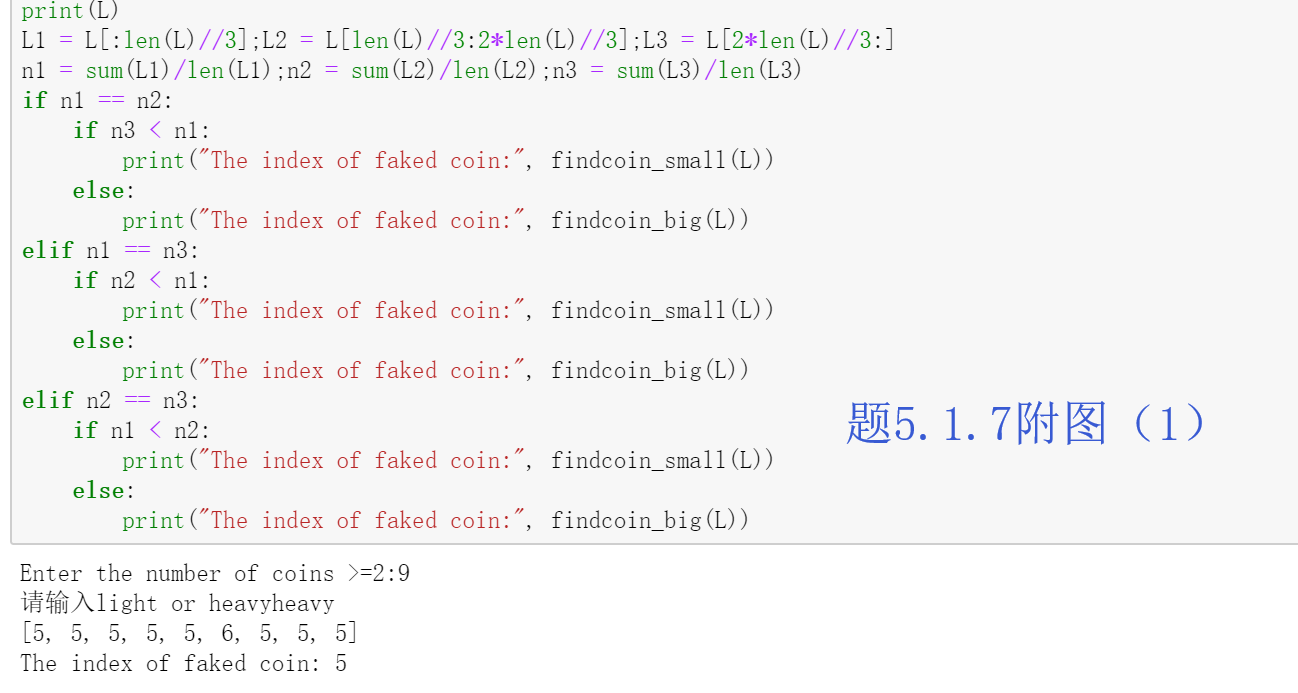
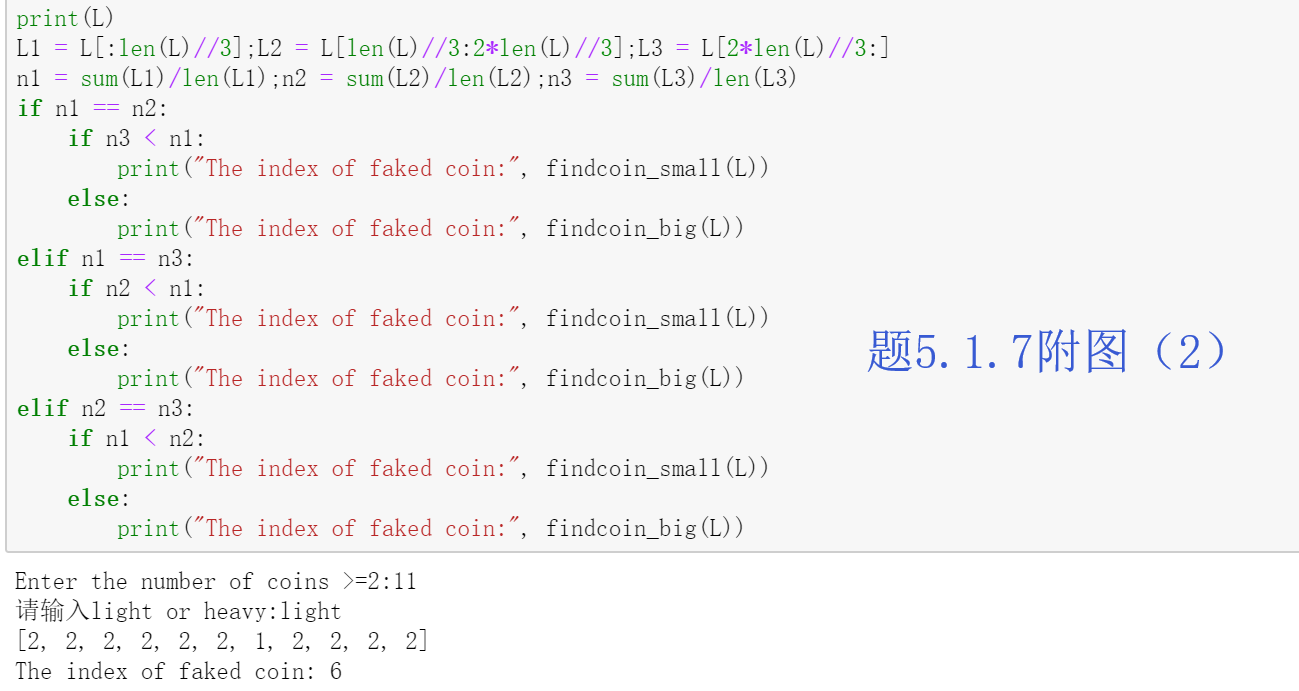
**print("The index of faked coin:", findcoin\_big(L))**

**elif n2 == n3:**

**if n1 < n2:**

**print("The index of faked coin:", findcoin\_small(L))**

**else:**

 **print("The index of faked coin:", findcoin\_big(L))**

1. **(10 points)改写Hanoi 程序， 假设A杆起初有n个圆盘， 从小到大编号为1，2，…, n，请打印出每次移动后A， B， C上的圆盘编号从小到大有哪些。请试验n=4.**

**count = 1**

**n = int(input("请输入盘子个数:"))**

**a = [i for i in range(1,n+1)]**

**b = []**

**c = []**

**def Hanoi(n,A,C,B):**

**global count,a,b,c**

**if n < 1:**

**print("False")**

**elif n == 1:**

**print ("%d:\t%s->%s"%(count,A,C),end=" ")**

**if A == "A" and C == "C":**

**c =[a.pop(0)] + c**

**if A == "C" and C == "A":**

**a =[c.pop(0)] + a**

**if A == "B" and C == "C":**

**c =[b.pop(0)] + c**

**if A == "C" and C == "B":**

**b =[c.pop(0)] + b**

**if A == "A" and C == "B":**

**b =[a.pop(0)] + b**

**if A == "B" and C == "A":**

**a =[b.pop(0)] + a**

**print(a,b,c)**

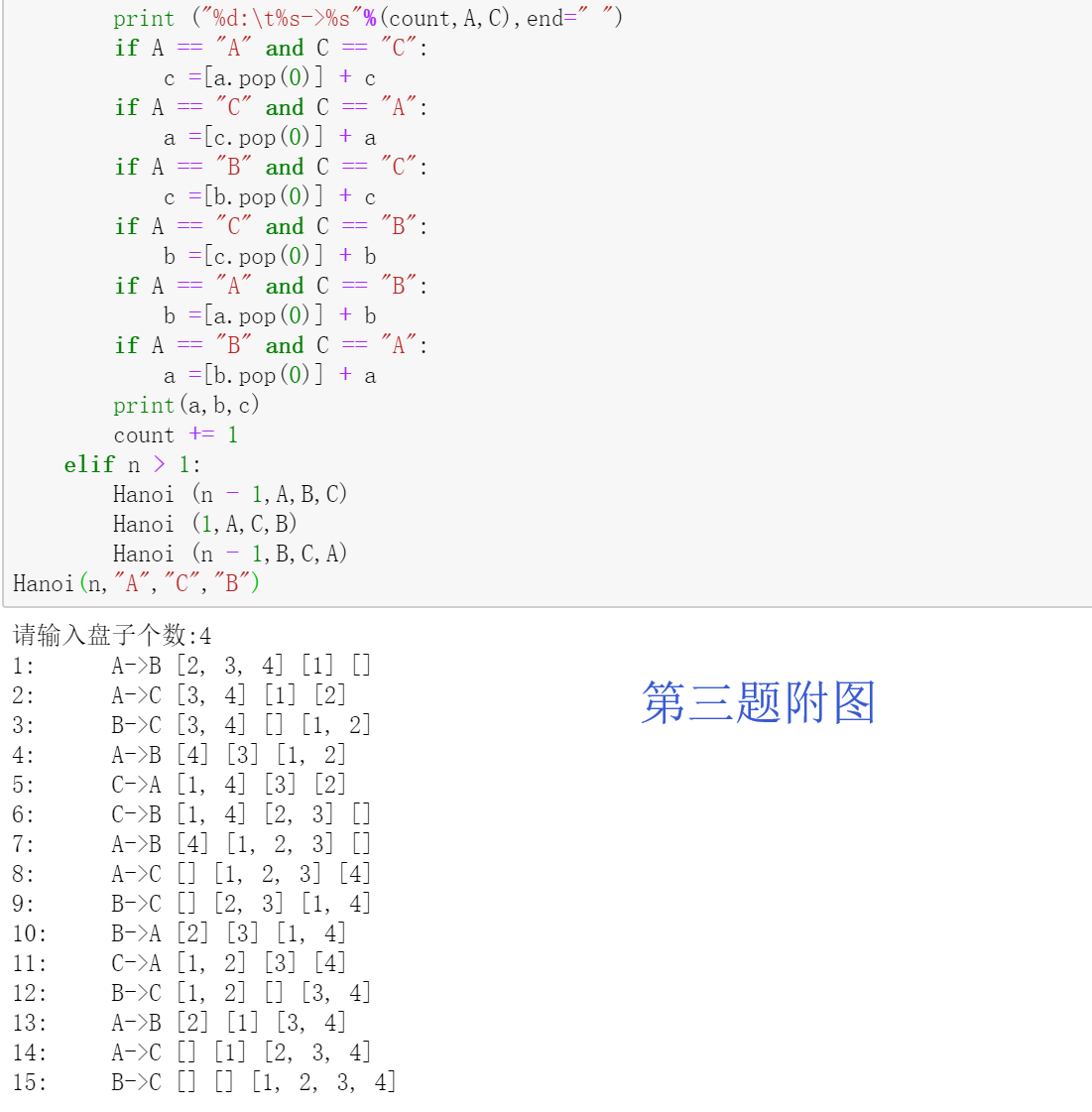
**count += 1**

**elif n > 1:**

**Hanoi (n - 1,A,B,C)**

**Hanoi (1,A,C,B)**

**Hanoi (n - 1,B,C,A)**

**Hanoi(n,"A","C","B")**

1. **（10 points）练习题 5.3.1, 5.3.2**

**5.3.1**

**可以计算计算总和，乘积和最大值，减法不可以，在某个运算符号后面的算式中加括号进行结合律时，扩号里面的符号不改变，才可以**

**例如 a+b+c=a+(b+c); a-b-c=a-(b+c)**

**前者没有改变，后者改变了，因此加号可以，减号不行**

**5.3.2**

**def M(L):**

**print(L)**

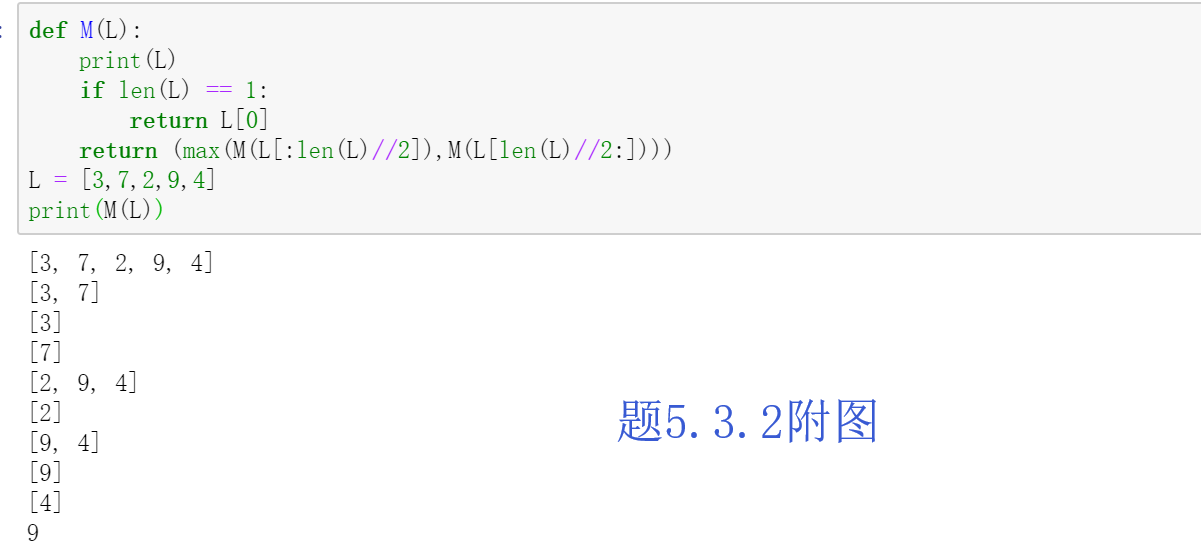
**if len(L) == 1:**

**return L[0]**

**return (max(M(L[:len(L)//2]),M(L[len(L)//2:])))**

**L = [3,7,2,9,4]**

**print(M(L))**



1. **（10 points）程序练习题 5.3.1, 5.3.2。 试验和解释。**

**程序练习题5.3.1：**

**def merge (L1,L2):**

**if len(L1) == 0:**

**return (L2)**

**if len(L2) == 0:**

**return (L1)**

**if L1[0] < L2[0]:**

**return ([L1[0]]+merge(L1[1:len(L1)],L2))**

**else:**

**return ([L2[0]]+merge(L1,L2[1:len(L2)]))**

**def msort(L):**

**k = len(L)**

**if k == 0 or k == 1:**

**return L**

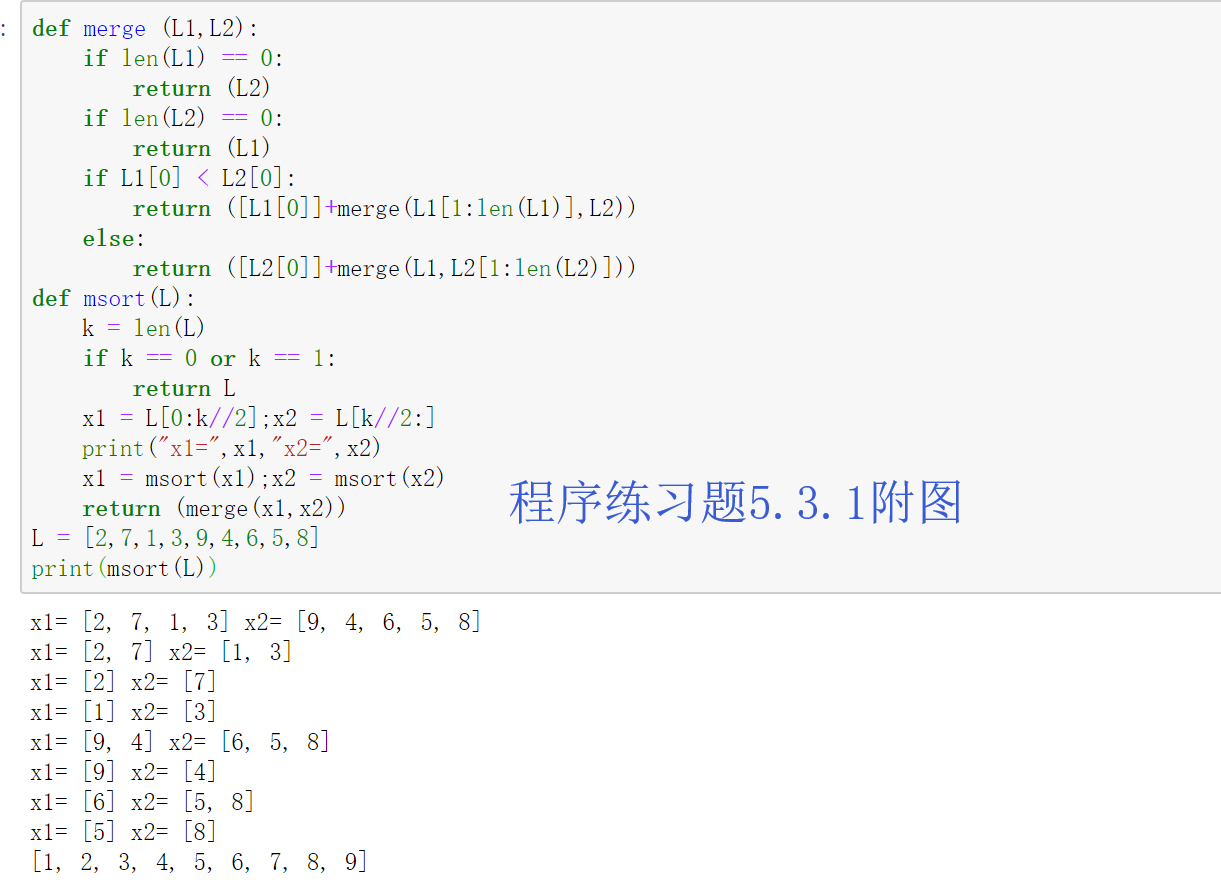
**x1 = L[0:k//2];x2 = L[k//2:]**

**print("x1=",x1,"x2=",x2)**

**x1 = msort(x1);x2 = msort(x2)**

**return (merge(x1,x2))**

**L = [2,7,1,3,9,4,6,5,8]**

**print(msort(L))**

**解释：merge函数是将两个有序的列表合并成一个新的有序的列表，msort函数首先将列表不断二分，使得生成一堆只有一个元素的子列表，然后通过比较两个相邻子列表元素的大小，合并成多个含有两个元素的有序列表，然后不断调用merge函数，不断合并，直到整个列表被有序还原**

**程序练习题5.3.2**

**def FA(a,b,c):**

**carry = (a and b) or (b and c) or (a and c)**

**sum1 =(a and b and c) or (a and (not b) and (not c)) or ((not a) and b and (not c))\**

**or ((not a) and (not b) and c)**

**return carry,sum1**

**def T(x):**

**if x == True:**

**return "1"**

**else:**

**return "0"**

**def add\_divide(x,y,c = False):**

**while len(x) < len (y):**

**x = [False] + x**

**while len(x) > len (y):**

**y = [False] + y**

**if len(x) == 1:**

**ctemp,stemp = FA(x[0],y[0],c)**

**return ctemp,T(stemp)**

**if len(x) == 0:**

**return c,[]**

**c1,s1 = add\_divide(x[len(x)//2:],y[len(y)//2:],c)**

**c2,s2 = add\_divide(x[:len(x)//2],y[:len(y)//2],c1)**

**return (c2,s2+s1)**

**def trans(x):**

**l=[]**

**for e in x:**

**if e == "1":**

**l.append(True)**

**else:**

**l.append(False)**

**return l**

**x1="101";x2="1100"**

**L1=trans(x1);L2=trans(x2)**

**print(L1,L2)**

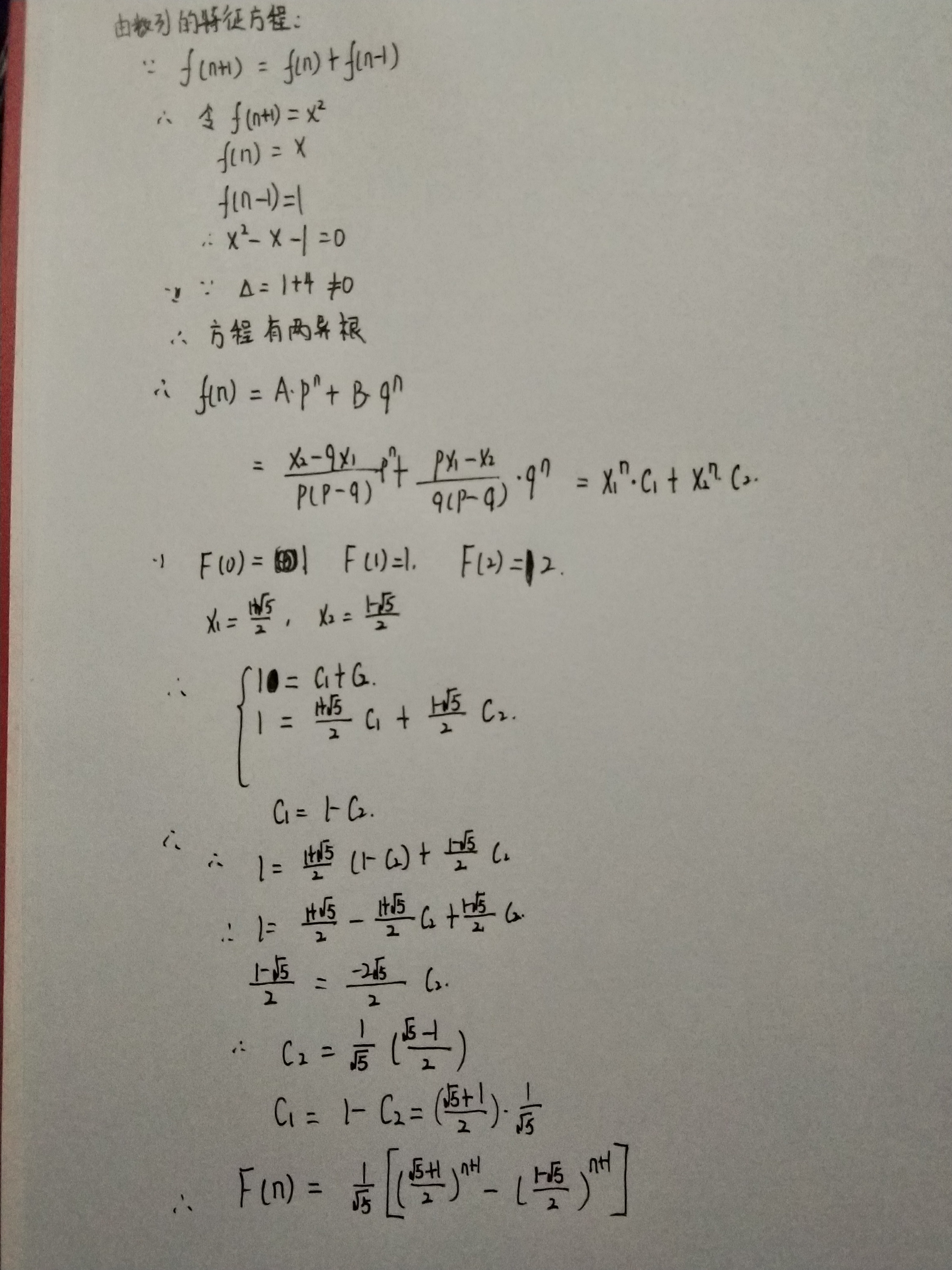
**a,b = add\_divide(L1,L2,c = False)**

**b = T(a) + b**

**print("Carry is:",a)**

**print("Sum is:",b)**

**解释：先将X,Y两个二进制数换成“False”&“True”组成的列表，然后对位数少的进行补0，也就是加False，然后就是不断二分，直到位数是1位时，进行carry的计算，并将carry的值传递给c1，carry为TRUE则表示进位，否则就是1+0或者0+0，无需进位，因为整个算法都是先计算len(x)//2的部分，因此是从最后一位往前计算的，s1，s2也就是各个位数的值，因为大小意义不同，所以采用了转化为字符串的相加，整个函数结束后，再根据最高位 的TRUE or False也就是是否进位来决定是否补1，所得到的值便是x+y的二进制值，而最后的carry就是x和y最高位的相加是否进位了**

1. **（10 points）F(0)=1; F(1)=1; F(n)=F(n-1)+F(n-2), n>=2. 请问 F(n)的 closed form是什么？提示，这个解肯定和Fibanacci number的解很像，只是系数不同罢了。**
2. **（5 points）关于学期末大项目，你们要设计一个游戏。建议你们参考我们的编程导论第六章，团队可以是2、3人，经过我的同意后，也可以是4人，但是4人的团队要想办法做个比较大的程序。我不希望一个人做，你们要学会如何与别人合作，好吗？请写下你们想要实现的游戏和现今团队的所有组员名单。这只是个计划，将来可以改变。**

**游戏：2048**

**成员：41-吴子靖，42-李泽浩，44-董辰尧**