**Introduction to Computer Science**

**Homework 7**

**沙行勉教授**

**Due Date: 星期二，Nov 27, 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\_Student ID\_your name\_your 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三章。有关函数调用栈帧的建立的课本练习题和习题，可以用教科书所写的简单方式，也可以用SEAL模拟器描述的较为实际的方式。请同学在答相关问题开始时，申明是用哪一个版本。**

1. **（每题5 points）3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, 3.4.6,**

**3.4.1 a,b,c均在函数外定义，a,b,c均为全局变量，d为局部变量**

**a=d\*c=b\*c=8**

**所以输出8，局部变量为d**

**3.4.2 func()中就有输出x的指令，之后又有输出a的指令**

**所以将输出两行10**

**3.4.3 a在函数外就被定义了，所以a不是局部变量，b是函数内的变量，也没有global函数修饰，所以b是局部变量，c=a+b=10+1=11,所以输出11**

**所以 a不是局部变量，b是局部变量，输出结果为11**

**3.4.4 执行do\_add函数，c=1是全局变量，此时的局部变量c1=a+b=5**

**再执行do\_sub函数，局部变量c2=c1-1=4**

**此时再执行do\_mul函数，全局变量c=c2\*c2=4\*4=16,输出此时的c=16，并将c=16返回给c2**

**然后执行do\_div函数，局部变量c3=c2/2=8,输出此时的c3=8，并将c3=8返回给c2**

**然后再回到do\_sub函数，再次输出c2=8，并将c2=8返回给c1**

**然后回到do\_add函数，再次输出c1=8**

**然后执行最后的print语句，输出全局变量c=16**

**所以最终输出结果为**

**16**

**8.0**

**8.0**

**8.0**

**16**

**3.4.5 a是一个定义在函数外的参数，他已经是一个全局变量了，不需要再用global来修饰，所以执行此函数会出现报错：a已经是一个全局变量了，请不要进行覆盖**

**3.4.6 a=10中的a是全局变量，函数内有语句a=a+10，左边的a是函数内的局部变量，而右边的a是全局变量，最终输出的是局部变量a，所以a（局部）=a（全局）+10，所以左边的a是没有定义的，于是出现了 local variable ‘a’ referenced before assignment**

1. **（每题8 points）Let ID=your student ID. Then, let k=ID %2. So k can be 0 or 1. Based on the value of your ID, you get value of k. Please tell us your k value.**

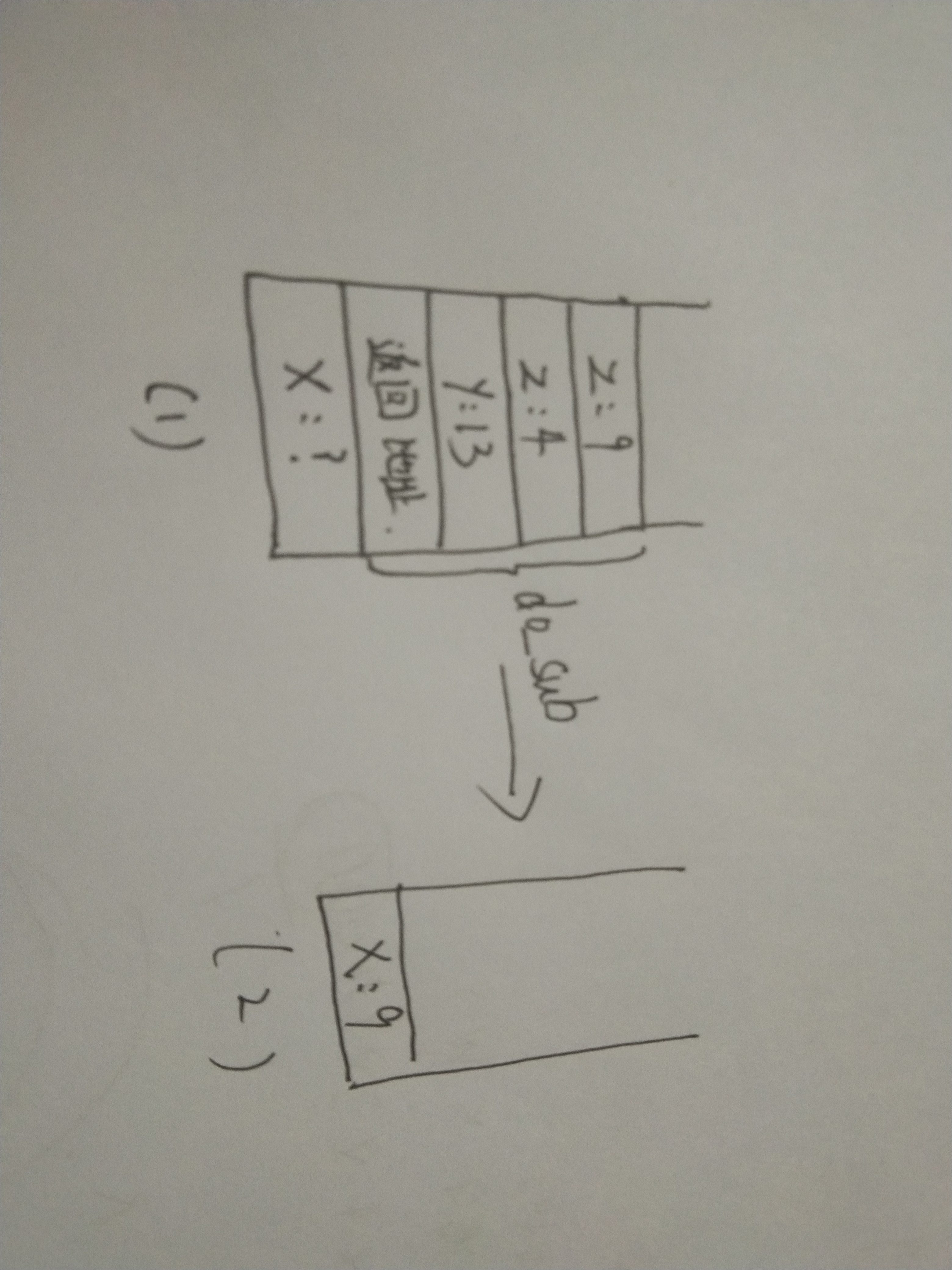
**If k==0: 做习题3.14， 3.16， 3.18， 3.20， 练习题 3.5.1  
If k==1: 做习题 3.15， 3.17， 3.19， 3.21， 练习题 3.5.2**

**K=1**

**3.15: b=a(局部)=a(全局)+b=40，print(a,b)为10,40**

**所以运行结果为10,40**

**3.17：因为栈的特点为“last in first out”在包含子函数的主函数系统中，子函数出现于主函数之后，却最先完成运算并返回主函数，与栈的特点恰好相符，所以我们用栈保存调用函数信息**

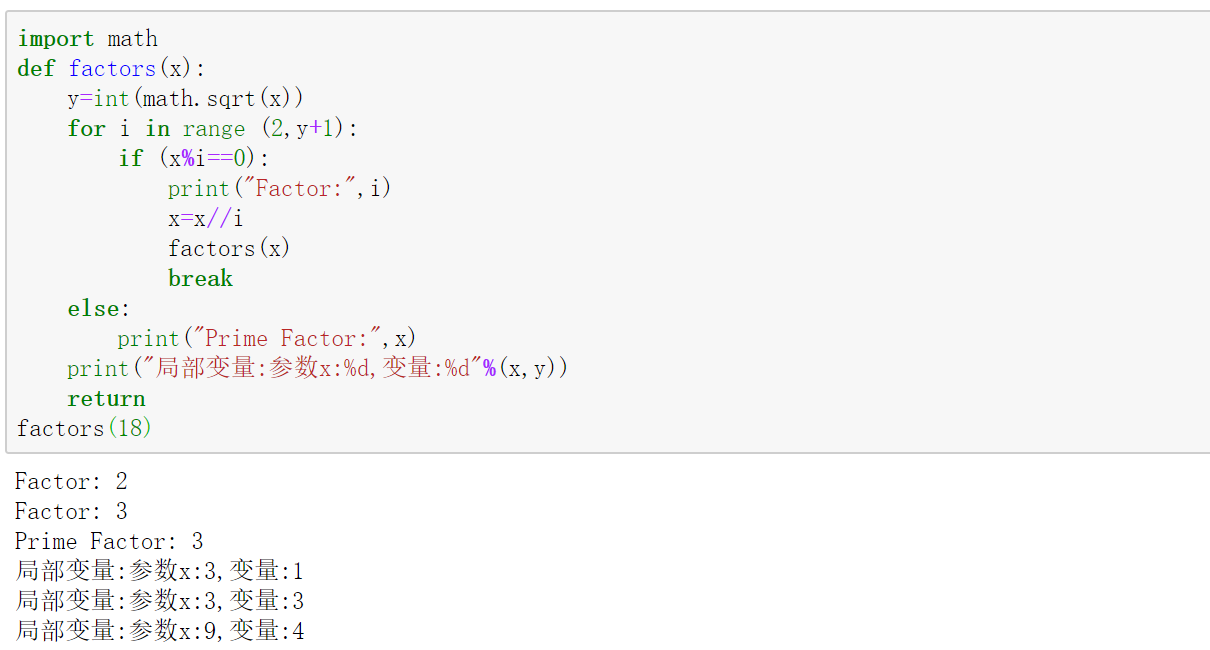
**3.19**

**3.21（1）z=11,x=11-5=6,print(x)=6，输出6**

**x=z-y=6,输出6**

**（2）不一样，第一个的y是在函数栈帧外，第二个的y在函数栈帧内，且第一个的x用global修饰了，不是局部变量，而第二个的x在函数栈帧内**

**3.5.2**



**3、（10 points）请完全按照SEAL的函数调用方式，描述主函数是如何调用子函数F()时，建立栈帧的过程，以及如何返回。请忠实写出SEAL的汇编代码，不要做任何简化。注意，F函数的x, y 是局部变量，在栈帧中是有位置的。  
def F(a,b):  
 x=1  
 y=a+b-x  
 return(y)**

**F(2, 3)**

**#<seal>**

**move R15,10000**

**move sp,R15**

**sub sp,sp,2**

**move R2,2**

**move R3,3**

**store -1(R15),R3**

**store -2(R15),R2**

**push R3**

**push R2**

**call Ladd**

**goto Lprint**

**Ladd:**

**push R15**

**move R15,sp**

**sub sp,sp,2**

**move R4,1**

**push R4**

**push R2**

**push R3**

**load R2,2(R15)**

**load R3,3(R15)**

**add R1,R2,R3**

**sub R1,R1,R4**

**store -1(R15),R1**

**Lreturn:**

**pop R3**

**pop R2**

**move sp,R15**

**pop R15**

**ret**

**Lprint:**

**\_pr R1**

**4、（20 points）请用SEAL写出如下的函数，使用汇编语言和函数标准栈的建立法方式。请完全按照SEAL中所描述的方式。可用SEAL中的\_pr打印。**

**def factors(x): #找到x的因数**

**y=x//2**

**for i in range(2,y+1):**

**if (x %i ==0): #发现i是x的因数**

**print("Factor:",i);**

**factors(x//i) #递归调用自己，参数变小是x//i**

**break #跳出for循环**

**else: #假如离开循环正常，没有碰到break，就执行else内的print，x是质数**

**print("Prime Factor:",x)**

p**rint("参数x:%d, 变量y:%d" %(x,y))**

**return**

**factors(18)**

**请在注释中写出你的Python对应代码，把你的全部代码拷贝到word报告中，并将程序的运行结果截图粘贴到报告中。**

**#<python:找因数>**

**#R0=18**

**#def factors(R0):**

**# R1=R0//2;R3=2**

**# R2=R0//R3**

**# R2=R2\*R3-R0**

**# while R2!=0:**

**# if R3<=R1:**

**# R3+=1**

**# R2=R0//R3**

**# R2=R2\*R3-R0**

**# else:**

**# break**

**# if R2==0：**

**# print("Factor:",R3)**

**# factors(R0//R3)**

**# if R2!=0:**

**# print("Prime Factor:",R0)**

**# print("参数R0:%d, 变量R1:%d" %(R0,R1))**

**# return**

**#<seal>**

**move R0,18**

**Lfactors:**

**div R1,R0,2**

**move R3,2**

**div R2,R0,R3**

**mul R2,R2,R3**

**sub R2,R2,R0**

**L2:**

**beqz R2,L3**

**sle R4,R3,R1**

**beqz R4,L3**

**add R3,R3,1**

**div R2,R0,R3**

**mul R2,R2,R3**

**sub R2,R2,R0**

**goto L2**

**L3:**

**beqz R2,L4**

**\_pr "Prime Factor:",R0**

**goto L5**

**L4:**

**\_pr "Factor:",R3**

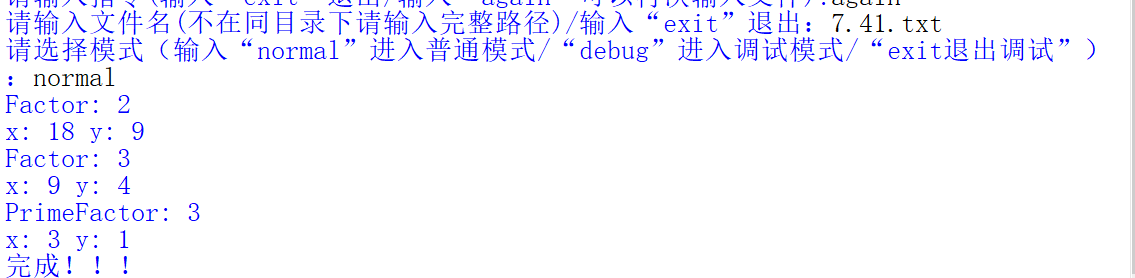
**\_pr "参数x:",R0,"参数y:",R1**

**div R0,R0,R3**

**goto Lfactors**

**L5:**

**\_pr "参数x:",R0,"参数y:",R1**



**5、（20 points）his programming assignment can be done by you alone or a group of two students. 写下合作同学的名字。**

**请用SEAL写出和执行汇编语言程序。输入一个字符串表示二进制数，以\_data的形式输入，放在地址0之后，此数组中的第一个数代表字符串的长度，然后是0或1的ASCII码：字符0是48，字符 1是49.。例如，“101”是用\_data 0, [3, 49, 48, 49]. 用最简单的方式计算出它的十进制数，将结果放在R1中，并打印。不准用mul, div指令。**

**提示：对于一个二进位数，如“1101”，它的十进位值是“110”的值乘以2后加1.如“1100”的值则是“110”的值乘以2后加0。所以同学可以从最最高位开始累计求值。另外，乘以2不过是左移一位罢了。**

**请在注释中写出你的Python对应代码，把你的全部代码拷贝到word报告中，并将程序的运行结果截图粘贴到报告中。**

**#<python>**

**#L=[3,49,48,49]**

**#R0=L[0]**

**#R1=1;R3=0**

**#while R1<=R0:**

**# if L[R1]==49:**

**# R2=1**

**# if L[R1]==48:**

**# R2=0**

**# R3=2\*R3+R2**

**# R1+=1**

**#print(R3)**

**#<seal>**

**\_data 0,[4,49,48,49,49]**

**move R1,1**

**move R3,0**

**load R0,(0)**

**L1:**

**sle R4,R1,R0**

**beqz R4,L2**

**load R5,0(R1)**

**sub R5,R5,48**

**beqz R5,L3**

**move R2,1**

**goto L4**

**L3:**

**move R2,0**

**L4:**

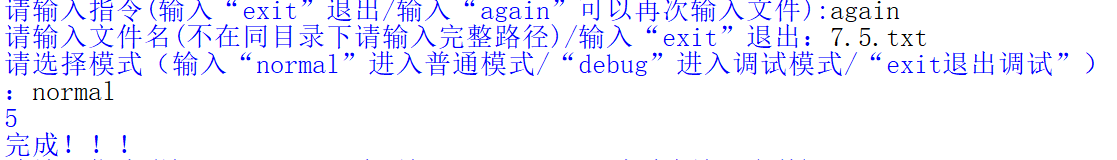
**shiftl R3,R3,1**

**add R3,R3,R2**

**add R1,R1,1**

**goto L1**

**L2:**

**\_pr R3**

**6、（20 points）his programming assignment can be done by you alone or a group of two students. 写下合作同学的名字。**

**请用SEAL完成排序程序。输入一个整数列表，以\_data的形式输入，放在地址10之后，此数组中的第一个数代表列表的长度n，然后是n个正整数。将排好序的列表以相同的形式放在输入列表之后，并打印。注意， \_pr 可以打印一连串主存内容。**

**例如， \_data 10, [20, 9, 12, 1, 100, 31, 5, 6, 29, 90, 10, 51, 21, 41, 23, 56, 64, 90, 98, 73, 73]。第一个数20代表后面有20个数。**

**你可以用任何排序算法。请在注释中写出你的Python对应代码，把你的全部代码拷贝到word报告中，并将程序的运行结果截图粘贴到报告中。**

**#<python>**

**#L=[20,9,12,1,100,31,5,6,29,90,10,51,21,41,23,56,64,90,98,73,73]**

**#R1=L[0]**

**#R2=R1**

**#R3=1;**

**#while R3<R2:**

**# R5=R3+1**

**# while R5<=R2:**

**# R4=L[R3]**

**# R6=L[R5]**

**# if R4>R6:**

**# L[R3]=R6**

**# L[R5]=R4**

**# R5+=1**

**# R3+=1**

**#print(L)**

**#<seal>**

**\_data 10,[20,9,12,1,100,31,5,6,29,90,10,51,21,41,23,56,64,90,98,73,73]**

**load R1,(10)**

**move R3,0**

**L1:**

**add R3,R3,1**

**slt R2,R3,R1**

**beqz R2,L2**

**add R5,R3,0**

**L3:**

**add R5,R5,1**

**sle R7,R5,R1**

**beqz R7,L1**

**load R4,10(R3)**

**load R6,10(R5)**

**slt R9,R6,R4**

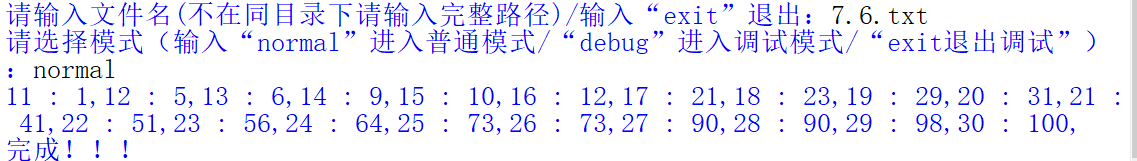
**beqz R9,L3**

**store 10(R3),R6**

**store 10(R5),R4**

**goto L3**

**L2:**

**\_pr (11)%20**