1.完成<计算机导论>习题 3.14,3.15,3.16,3.19,3.21,题目中关于栈帧的回答可以用书上的方式，也可以用 SEAL 的方式。

3.14

```
40 40
```

3.15

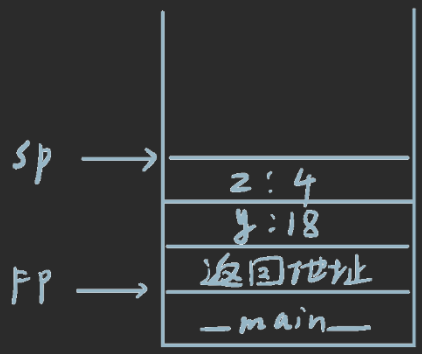
```
10 40
```

3.16

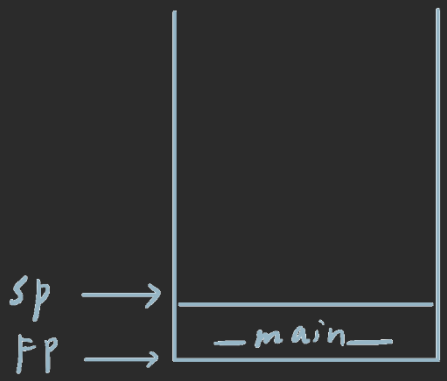
```
20
30 15
```

3.19

(1)



(2)



3.21

(1)

函数一：

```
6
```

函数二：

```
6
```

（2）

不相同。因为函数一的调用函数后的栈帧中不会包含全局变量 x，y（全局变量全部存储在内存的静态区中，而不是在堆或者栈）；而函数二的调用函数后的栈帧中会有局部变量压入栈里，即应当由 x，y，z 三个变量在栈中。

2.请用 SEAL 写出和执行汇编语言程序：输入二个正整数，放在寄存器 R0 和 R2 中，比较大小，将结果放在 R1 中，并打印。如果 R0=R2,R1=1;如果 R0<R2,R1=0;如果 R0>R2,R1=2.例如 mov R0,10;movR2,12;结果是 0.又例如 mov Ro,10;movR2,10;结果是 1.

```
# <Python Code>
# a = 10
# b = 7
# if a < b:
#     ret = 0
# elif a <= b:
#     ret = 1
# else:
#     ret = 2
# print(ret)

mov R0,10
mov R2,7
slt R3,R0,R2
beqz R3,L1
mov R1,0
goto L3
L1:
sle R3,R0,R2
beqz R3,L2
mov R1,1
goto L3
L2:
mov R1,2
L3:
_pr R1
```

Shell:

```
请输入文件名(不在同目录下请输入完整路径)/输入“exit”退出：hw7_2.txt
请选择模式（输入“normal”进入普通模式/“debug”进入调试模式/“exit退出调试”）：normal
2
完成！！！
```

若将前两句改为

```
mov R0,7
mov R2,7
```

则 Shell：

```
请输入文件名(不在同目录下请输入完整路径)/输入“exit”退出：hw7_2.txt
请选择模式（输入“normal”进入普通模式/“debug”进入调试模式/“exit退出调试”）：normal
1
完成！！！
```

若将前两句改为

```
mov R0,5
mov R2,7
```

则 Shell：

```
请输入文件名(不在同目录下请输入完整路径)/输入“exit”退出：hw7_2.txt
请选择模式（输入“normal”进入普通模式/“debug”进入调试模式/“exit退出调试”）：normal
0
完成！！！
```

3.请用 SEAL 写出和执行汇编语言程序：输入一个正整数，放在寄存器 R0 中，请用简单的方式计算出它的二进制数有多少个 1,将结果放在 R1 中，并打印。例如 move R0,13,结果是 3.

```
# <Python Code>
# a = 13
# c = 0
# d = 0
# while a > 0:
#     d = a >> 1
#     d = d << 1
#     d = a - d
#     if d == 1:
#         c += 1
#     a = a >> 1
# print(c)

mov R0,13
mov R1,0
mov R2,0

L2:
slt R4,R1,R0
beqz R4,L3
shiftr R3,R0,1
shiftl R3,R3,1
sub R3,R0,R3
sle R4,R3,R1
beqz R4,L0
goto L1
L0:
add R2,R2,1
L1:
shiftr R0,R0,1
goto L2
L3:
_pr R2
```

Shell:

```
请输入文件名(不在同目录下请输入完整路径)/输入“exit”退出：hw7_3.txt
请选择模式（输入“normal”进入普通模式/“debug”进入调试模式/“exit退出调试”）：normal
3
完成！！！
```

4.请用 SEAL 写出和执行汇编语言程序：输入一个整数列表，计算整数列表中为偶数的数值和，以 data 的形式输入，此数组中的第个数代表列表的长度，然后是列表的整数值，例如整数列表[5,2,8,11,31,25,101],在_ data1 的形式是：_data1,[7,5,2,8,11,31,25,101].第一个数 7 代表后面有 7 个整数，因为列表中的偶数是 2 与 8,所以它们的和是 10,将结果放在 R1 中，并打印

```
# <Python Code>
# L1 = [7, 5, 2, 8, 11, 31, 25, 101]
# a = 0
# for i in L1:
#     if i % 2 == 0:
#         a += i
# print(a)

_data 1,[7,5,2,8,11,31,25,101]

mov R1,0
mov R2,0
mov R0,1
load R3,0(R0)
add R0,R0,1
L0:
slt R4,R2,R3
beqz R4,L2
load R5,0(R0)
shiftr R6,R5,1
shiftl R6,R6,1
sub R6,R5,R6
sle R4,R6,0
beqz R4,L3
add R1,R1,R5
L3:
add R2,R2,1
add R0,R0,1
goto L0
L2:
_pr R1
```

Shell:

```
请输入文件名(不在同目录下请输入完整路径)/输入“exit”退出：hw7_4.txt
请选择模式（输入“normal”进入普通模式/“debug”进入调试模式/“exit退出调试”）：normal
10
完成！！！
```

5. [hint:此题关键在于递归函数栈帧是如何建立和返回的]用 SEAL 写出如下的函数，请完全按照 SEAL 中所描述的函数栈帧建立的方式，可用 SEAL 中的_pr 打印

```
def factors(x):
    y = x // 2
    for i in range(2, y + 1):
        if x % i == 0:
            print("Factor:", i)
            factors(x // i)
            break
    else:
        print("Prime Factor:", x)
    print("参数x: %d, 变量y: %d" % (x, y))
    return


factors(18)
```

```
mov R15,1000  # 初始化 fp，地址为 1000
mov sp,R15
sub sp,sp,1

mov R1,18 # x = 18
store -1(R15),R1

call Lfactors
goto Lend

################################factors 函数开始
Lfactors:
push R15
mov R15,sp


push R1
push R2

load R1,2(R15)
shiftr R2,R1,1   # y = x // 2

mov R3,2      # i = 2

Lloop:
sle R4,R3,R2
beqz R4,Lout_else

##########编写除法运算##########
mov R5,0      # 除数
```

```
mov R6,0      # 余数
mov R7,R1
Lloop1:
sle R4,R3,R7
beqz R4,Lout1
sub R7,R7,R3
add R5,R5,1
goto Lloop1
Lout1:
##########除法运算结束##########

mov R6,R7 # 取余
###################################
sle R4,R6,0   # if x 5% i == 0:
beqz R4,L0
_pr R3

sub sp,sp,1
store 0(sp),R5

call Lfactors

goto Lout
###################################
goto Lout_else
L0:
add R3,R3,1
goto Lloop

Lout_else:
_pr R1

Lout:
_pr R1,R2
goto Lreturn

Lreturn:

pop R2
pop R1
mov sp,R15
pop R15
ret
################################factors 函数结束

Lend:
# 这是空语句
```

Shell:

```
2
3
3
3 1
9 4
18 9
```