1. Do the following exercises from Chapter 7. Note that the questions ask you to draw diagrams. For these questions, you will need to either use a drawing program or answer the problems with paper and pencil and scan your solution to a PDF. Each exercise is worth 10 points.

- Exercise 7.7: Using Figure 7.5 as your model, illustrate the execution of function call f(1) as well as the state of the program stack. Function f() is defined in module stack.py.

```
1  def h(n):
2      print('Start h')
3      print(1/n)
4      print(n)
5
6  def g(n):
7      print('Start g')
8      h(n-1)
9      print(n)
10
11 def f(n):
12     print('Start f')
13     g(n-1)
14     print(n)
```

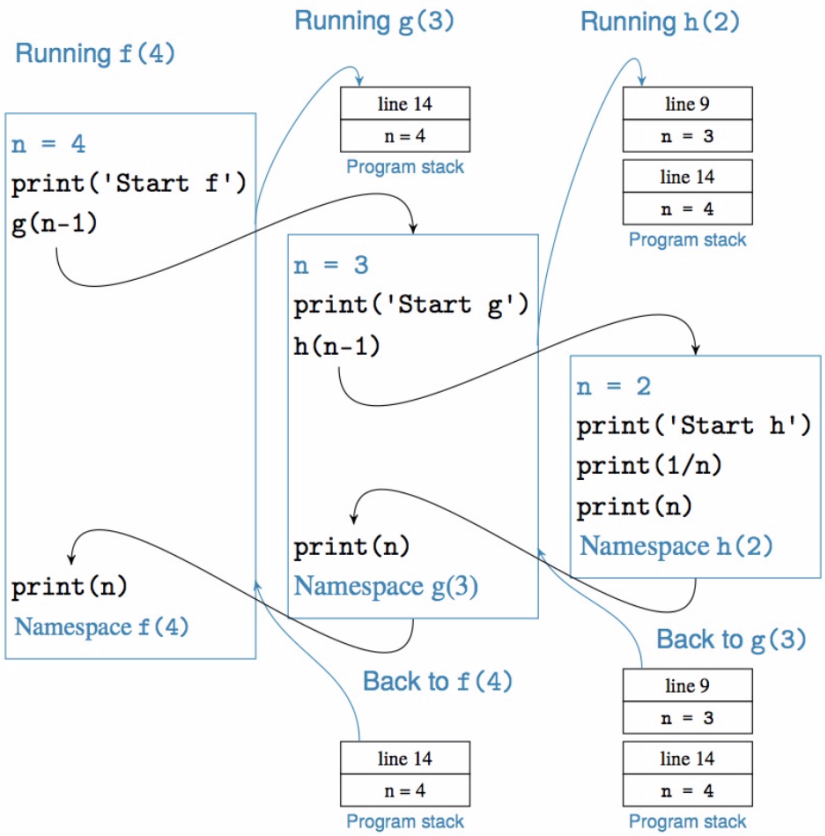

Figure 7.5 Execution of `f(4)`, part 2. The function call `f(4)` executes in its own namespace. When function call `g(3)` is made, the namespace of `f(4)` is pushed onto the program stack. The call `g(3)` runs in its own namespace. When the call `h(2)` is made, the namespace of `g(3)` is also pushed onto the program stack. When function call `h(2)` terminates, the namespace of `g(3)` is restored by popping the top stack frame of the program stack; its execution continues from the line stored in the stack frame (i.e., line 9). When `g(3)` terminates, the namespace of `f(4)` and its execution are restored by popping the program stack again.

- Exercise 7.10: This exercise relates to modules one, two, and three:

```
import two

def f1():
    two.f2()

def f4():
    print('Hello!')
```

```
import three

def f2():
    three.f3()
```

```
import one

def f3():
    one.f4()
```

When module one is imported into the interpreter shell, we can execute f1():
>>> import one
>>> one.f1()
Hello!

(For this to work, list sys.path should include the folder containing the three modules.) Using Figure 7.13 as your model, draw the namespaces corresponding to the three imported modules and also the shell namespace. Show all the names defined in the three imported namespaces as well as the objects they refer to.

- Exercise 7.11: After importing one in the previous problem, we can view the attributes of one:

>>> dir(one)
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'f1', 'f4', 'two']

However, we cannot view the attributes of two in the same way:

```
>>> dir(two)
Traceback (most recent call last):
  File "<pyshell#202>", line 1, in <module>
    dir(two)
NameError: name 'two' is not defined
```

Why is that? Note that importing module one forces the importing of modules two and three. How can we view their attributes using function dir()?

- Exercise 7.14: Let list 1st be:

```
>>> lst = [2,3,4,5]
```

Translate the next list method invocations to appropriate calls to functions in namespace list:

(a) lst.sort()

(b) lst.append(3)

(c) lst.count(3)

(d) lst.insert(2, 1)

2. Solve the following programming problems by implementing the functions in the skeleton file HW6.py. Each problem is worth 20 points.

- Problem 6.31. Note that this problem requires you to implement 2 functions.
    - Craps is a dice-based game played in many casinos. Like blackjack, a player plays against the house. The game starts with the player throwing a pair of standard, six-sided dice. If the player rolls a total of 7 or 11, the player wins. If the player rolls a total of 2, 3, or 12, the player loses. For all other roll values, the player will repeatedly roll the pair of dice until either she rolls the initial value again (in which case she wins) or 7 (in which case she loses)
        a) Implement function craps() that takes no argument, simulates one game of craps, and returns 1 if the player won and 0 if the player lost

            >>> craps()

            0

            >>> craps()

            1

            >>> craps()

            1

        b) Implement function testCraps() that takes a positive integer *n* as input, simulates *n* games of craps, and returns the fraction of games the player won

            >>> testCraps(10000)

            0.4844

            >>> testCraps(10000)

            0.492

- Problem 6.29 – But change the function so that it RETURNS a list of the social networks instead of printing them. For example, in the sample execution given on page 198, the function should RETURN [{0, 1, 2}, {3, 4}]. Note that neither the order in which the numbers appear in the sets nor the order in which the sets appear in the list matters. The list [{4, 3}, {1, 0, 2}] would also be a valid answer.
    - In your class, many students are friends. Let's assume that two students sharing a friend must be friends themselves; in other words, if students 0 and 1 are friends and students 1 and 2 are friends, then students 0 and 2 must be friends. Using this rule, we can partition the students into circles of friends.
    - To do this, implement function networks() that takes two input arguments. The first is the number of *n* of students. We assume students are identified using integers 0 through *n* – 1. The second input argument is a lsit of tuple objects that define friends. For example, tuple (0,2) defines students 0 and 2 as friends. Function networks() should print the partition of students into circles of friends as illustrated:

      >>> networks(5, [(0, 1), (1, 2), (3, 4)])

      Social network 0 is {0, 1, 2}

      Social network 1 is {3, 4}

- Problem 7.16 – Note that this problem refers to problem 6.35 which you were not assigned previously, but which you basically need to solve in order to do 7.16. This problem does print messages to the screen. It does not return anything.
    - Develop a simple game that teaches kindergarteners how to add single-digit numbers.
    - Your function game() will take an integer n as input and then ask n single-digit addition questions. The numbers to be added should be chosen randomly from the range [0, 9] (i.e., 0 to 9 inclusive). The user will enter the answer when prompted. Your function should print 'Correct' for correct answers and 'Incorrect' for incorrect answers. After n questions, your function should print the number of correct answers.

      >>> game(3)

      8+2=

      Enter answer: 10 Correct. 6+7=

      Enter answer: 12 Incorrect. 7+7= Enter answer: 14 Correct.

      You got 2 correct answers out of 3