**Starting Out with Python 4e (Gaddis)**

**Chapter 12 Recursion**

**TRUE/FALSE**

1. A recursive function must have some way to control the number of times it repeats.

ANS: T

2. In many cases it is easier to see how to solve a problem with recursion than with a loop.

ANS: F

3. If a recursive solution is evident for a particular problem, and if the recursive algorithm does not slow system performance by an intolerable amount, then recursion would probably be a good design choice.

ANS: T

4. A base case is not necessary for all recursive algorithms.

ANS: F

5. There must be only one function involved in any recursive solution.

ANS: F

6. Each time a function is called in a recursive solution, the system incurs overhead that is not incurred with a loop.

ANS: T

7. When, in a recursive solution, function A calls function B which, in turn, calls function A, this is known as indirect recursion.

ANS: T

8. A problem can normally be solved with recursion if it can be broken down into smaller problems that are identical in structure to the overall problem.

ANS: T

9. Recursive algorithms are always more concise and efficient than iterative algorithms.

ANS: F

10. Recursion is sometimes required to solve certain types of problems.

ANS: F

**MULTIPLE CHOICE**

1. In a recursive solution, if the problem cannot be solved now, then a recursive function reduces it to a smaller but similar problem and

|  |  |
| --- | --- |
| a. | exits |
| b. | returns to the main function |
| c. | returns to the calling function |
| d. | calls itself to solve the smaller problem |

ANS: D

2. What is the first step to take in order to apply a recursive approach?

|  |  |
| --- | --- |
| a. | Identify at least one case in which the problem can be solved without recursion. |
| b. | Determine a way to solve the problem in all circumstances using recursion. |
| c. | Identify a way to stop the recursion. |
| d. | Determine a way to return to the main function. |

ANS: A

3. What is the second step to take in order to apply a recursive approach?

|  |  |
| --- | --- |
| a. | Identify at least one case in which the problem can be solved without recursion. |
| b. | Determine a way to use recursion to solve the problem in all circumstances which cannot be solved without recursion. |
| c. | Determine a way to return to the main function. |
| d. | Identify a way to stop the recursion. |

ANS: B

4. If, in a recursive solution, function A calls function B which calls function C, this is called \_\_\_\_\_\_\_\_\_\_ recursion.

|  |  |
| --- | --- |
| a. | continuous |
| b. | direct |
| c. | three function call |
| d. | indirect |

ANS: D

5. A problem can be solved with recursion if it can be broken down into \_\_\_\_\_\_\_\_\_\_ problems.

|  |  |
| --- | --- |
| a. | smaller |
| b. | one-line |
| c. | manageable |
| d. | modular |

ANS: A

6. The base case is the case in which the problem can be solved without

|  |  |
| --- | --- |
| a. | loops |
| b. | decisions |
| c. | objects |
| d. | recursion |

ANS: D

7. If a problem can be solved immediately without recursion, then the recursive function

|  |  |
| --- | --- |
| a. | solves it and returns |
| b. | exits |
| c. | returns a default value |
| d. | generates a run-time error |

ANS: A

8. The process of calling a function requires

|  |  |
| --- | --- |
| a. | a slow memory access |
| b. | a quick memory access |
| c. | several actions to be performed by the computer |
| d. | one action to be performed by the computer |

ANS: C

9. Which of the following describes the base case in a recursive solution?

|  |  |
| --- | --- |
| a. | a case in which the problem can be solved without recursion |
| b. | the case in which the problem is solved through recursion |
| c. | the way to stop the recursion |
| d. | the way to return to the main function |

ANS: A

10. Recursion is

|  |  |
| --- | --- |
| a. | never required to solve a problem |
| b. | required to solve certain mathematical problems |
| c. | sometimes required to solve string problems |
| d. | required to solve some problems |

ANS: A

11. A function is called from the main function for the first time and then calls itself seven times. What is the depth of recursion?

|  |  |
| --- | --- |
| a. | 8 |
| b. | 2 |
| c. | 1 |
| d. | 7 |

ANS: D

12. What defines the depth of recursion?

|  |  |
| --- | --- |
| a. | the length of the algorithm |
| b. | the number of function calls |
| c. | the number of times the function calls itself |
| d. | the number of times the function goes to the base case |

ANS: C

13. Recursive functions are \_\_\_\_\_\_\_\_\_\_ iterative algorithms.

|  |  |
| --- | --- |
| a. | more efficient than |
| b. | less efficient than |
| c. | as efficient as |
| d. | impossible to compare to |

ANS: B

14. A recursive function includes \_\_\_\_\_\_\_\_\_\_ which are not necessary in a loop structure.

|  |  |
| --- | --- |
| a. | function calls |
| b. | conditional clauses |
| c. | overhead actions |
| d. | object instances |

ANS: C

15. Which would be the base case in a recursive solution to the problem of finding the factorial of a number. Recall that the factorial of a non-negative whole number is defined as *n!* where:

*If n = 0, then n! = 1*

*If n > 0, then n! = 1 x 2 x 3 x ... x n*

|  |  |
| --- | --- |
| a. | *n = 0* |
| b. | *n = 1* |
| c. | *n > 0* |
| d. | The factorial of a number cannot be solved with recursion. |

ANS: A

**COMPLETION**

1. All the cases of a recursive solution other than the base case are called the \_\_\_\_\_\_\_\_\_\_ case.

ANS: recursive

2. The base case does not require \_\_\_\_\_\_\_\_\_\_, so it stops the chain of recursive calls.

ANS: recursion

3. Recursive function calls are \_\_\_\_\_\_\_\_\_\_ efficient than loops.

ANS: less

4. Each time a function is called, the system incurs \_\_\_\_\_\_\_\_\_\_ that is not necessary with a loop.

ANS: overhead

5. A solution using a(n) \_\_\_\_\_\_\_\_\_\_ is usually more evident than a recursive solution.

ANS: loop

6. A function is called from the main function and then it calls itself five times. The depth of recursion is \_\_\_\_\_\_\_\_\_\_.

ANS: five

7. The majority of repetitive programming tasks are best done with \_\_\_\_\_\_\_\_\_\_\_.

ANS: loops

8. A recursion in which a function directly calls itself is known as \_\_\_\_\_\_\_\_\_\_\_ recursion.

ANS: direct

9. Usually a problem solved by recursion is reduced by making the value of one or more parameters \_\_\_\_\_\_\_\_\_\_ with each recursive call.

ANS: smaller

10. Some problems are more \_\_\_\_\_\_\_\_\_\_ solved with recursion than with a loop.

ANS: easily