CMPT 1105 - Review Questions 12

Multiple Choice

1. A recursive function _____.

    a. calls a different function

    b. abnormally halts the program

    c. calls itself

    d. can only be called once

2. A function is called once from a program's main function, then it calls itself four times. The depth of recursion is _____.

    a. one

    b. four

    c. five

    d. nine

3. The part of a problem that can be solved without recursion is the _____ case.

    a. base

    b. solvable

    c. known

    d. iterative

4. The part of a problem that is solved with recursion is the _____ case.

    a. base

    b. iterative

    c. unknown

    d. recursion

5. When a function explicitly calls itself, it is called _____ recursion.

    a. explicit

    b. modal

    c. direct

    d. indirect

6. When function A calls function B, which calls function A, it is called _____ recursion.

    a. implicit

    b. modal

    c. direct

    d. indirect

7. Any problem that can be solved recursively can also be solved with a _____.

    a. decision structure

    b. loop

c. sequence structure

d. case structure

8. Actions taken by the computer when a function is called, such as allocating memory for parameters and local variables, are referred to as _____.

a. overhead

b. set up

c. clean up

d. synchronization

9. A recursive algorithm must _____ in the recursive case.

a. solve the problem without recursion

b. reduce the problem to a smaller version of the original problem

c. acknowledge that an error has occurred and abort the program

d. enlarge the problem to a larger version of the original problem

10. A recursive algorithm must _____ in the base case.

a. solve the problem without recursion

b. reduce the problem to a smaller version of the original problem

c. acknowledge that an error has occurred and abort the program

d. enlarge the problem to a larger version of the original problem

True or False

1. An algorithm that uses a loop will usually run faster than an equivalent recursive algorithm.

2. Some problems can be solved through recursion only.

3. It is not necessary to have a base case in all recursive algorithms.

4. In the base case, a recursive method calls itself with a smaller version of the original problem.

Short Answer

1. In Program 12-2, presented earlier in this chapter, what is the base case of the message function?

2. In this chapter, the rules given for calculating the factorial of a number are as follows:

If n=0, then factorial(n) = 1

If n > 0 then factorial(n) = n X factorial(n - 1)

If you were designing a function from these rules, what would the base case be? What would the recursive case be?

3. Is recursion ever required to solve a problem? What other approach can you use to solve a problem that is repetitive in nature?

4. When recursion is used to solve a problem, why must the recursive function call itself to solve a smaller version of the original problem?

5. How is a problem usually reduced with a recursive function?

Algorithm Workbench

1. What will the following program display?

```
def main():
      num = 0
      show_me(num)
def show_me(arg):
      if arg < 10:
            show_me(arg + 1)
      else:
            print(arg)
main()
```

2. What will the following program display?

```
def main():
      num = 0
      show_me(num)
def show_me(arg):
      print(arg)
      if arg 10:
            show_me(arg + 1)
main()
```

3. The following function uses a loop. Rewrite it as a recursive function that performs the same operation.

```
def traffic_sign(n):
      while n > 0:
            print('No Parking')
            n = n - 1
```

Programming Exercises

1. Recursive Printing

Design a recursive function that accepts an integer argument, n, and prints the numbers 1 up through n.

2. Recursive Multiplication

Design a recursive function that accepts two arguments into the parameters x and y. The function should return the value of x times y. Remember, multiplication can be performed as repeated addition as follows:

$$7 \times 4 = 4 + 4 + 4 + 4 + 4 + 4 + 4$$

(To keep the function simple, assume x and y will always hold positive nonzero integers.)

3. Recursive Lines

Write a recursive function that accepts an integer argument, n. The function should display n lines of asterisks on the screen, with the first line showing 1 asterisk, the second line showing 2 asterisks, up to the nth line which shows n asterisks.

4. Sum of Numbers

Design a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4, . . . 50. Use recursion to calculate the sum.

5. Recursive Power Method

Design a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised, and the exponent. Assume the exponent is a nonnegative integer.