AP Computer Science
Chapter 12 Test

1 Multiple-Choice Questions

1) The following method should return true if the `int` parameter is even and either positive or 0, and false otherwise. Which set of code should you use to replace ... so that the method works appropriately?

```
        public boolean question1(int x)  { ... }
```

A) `if (x = = 0) return true;else if (x < 0) return false;else return question3(x - 1);`
B) `if (x = = 0) return false;else if (x < 0) return true;else return question1(x - 1);`
C) `if (x = = 0) return true;else if (x < 0) return false;else return question1(x - 2);`
D) `if (x = = 0) return false;else if (x < 0) return true;else return question1(x - 2);`
E) `return(x = = 0);`

Answer: C

For the questions#2 ~ 5 below, refer to the following recursive factorial method.

```
        public int factorial(int x)
        {
              if (x > 1) return x * factorial (x - 1);
              else return 1;
        }
```

2) What is returned if `factorial(3)` is called?
A) 0
B) 1
C) 3
D) 6
E) 9

Answer: D

3) What is returned if `factorial(0)` is called?
A) 0
B) 1
C) 2
D) nothing, `factorial(0)` causes infinite recursion
E) nothing, `factorial(0)` produces a run-time error

Answer: B

4) How many times is the factorial method invoked if originally called with `factorial(5)`? Include the original method call in your counting.
A) 1
B) 4
C) 5

D) 6
E) 7

Answer: C

5) What condition defines the base case for this method?
A) (`x > 1`)
B) (`x = = 1`)
C) (`x = = 0`)
D) (`x <= 0`)
E) (`x <= 1`)

Answer: E

6) What is wrong with the following recursive sum method? The method is supposed to sum up the values between 1 and `x` (for instance, `sum(5)` should be 5 + 4 + 3 + 2 + 1 = 15).

```
public int sum(int x)
{
      if (x = = 0) return 0;
      else return sum(x - 1) + x;
}
```

A) the base case should return 1 instead of 0
B) the recursive case should return `sum(x - 1) + 1`; instead of `sum(x - 1) + x`;
C) the base case condition should be (`x <= 0`) instead of (`x = = 0`)
D) the recursive case should return `sum(x) + 1`;
E) the method should return a `boolean` instead of an `int`

Answer: C

7) What does the following method compute? Assume the method is called initially with `i = 0`

```
public int question9(String a, char b, int i)
{
      if (i = = a.length( )) return 0;
      else if (b = = a.charAt(i)) return question9(a, b, i+1) + 1;
      else return question9(a, b, i+1);
}
```

A) the length of `String a`
B) the length of `String` a concatenated with `char b`
C) the number of times `char b` appears in `String a`
D) returns 1 if char `b` appears in `String a` at least once, and 0 otherwise
E) the char which appears at location `i` in `String a`

Answer: C

For the questions #8 ~12 below, assume that int[ ] a = {6, 2, 4, 6, 2, 1, 6, 2, 5} and consider the two recursive methods below foo and bar.

```
public int foo(int[ ] a, int b, int j) {
      if (j < a.length)
          if (a[j] != b) return foo (a, b, j+1);
          else return foo (a, b, j+1) + 1;
```

```
            else return 0;
        }

        public int bar(int[ ] a, int j) {
            if (j < a.length)
                return a[j] + bar(a, j+1);
            else return 0;
        }
```

8) What is the result of calling `foo(a, 2, 0)`;?
A) 0
B) 1
C) 2
D) 3
E) 4

Answer: D

9) What is the result of calling `foo(a, 3, 0)`;?
A) 0
B) 1
C) 2
D) 3
E) 4

Answer: A

10) What is the result of calling `foo(a, 2, 9)`;?
A) 0
B) 1
C) 2
D) 3
E) 4

Answer: A

11) What is the result of calling `bar(a, 0)`;?
A) 0
B) 5
C) 6
D) 12
E) 34

Answer: E

12) What is the result of `bar(a, 8)`;?
A) 0
B) 5
C) 6
D) 12
E) 34

Answer: B

13) What does the following recursive method determine?

```
        public boolean question13(int[ ]a, int[ ] b, int j)
        {
              if (j = = a.length) return false;
              else if (j = = b.length) return true;
              else return question13(a, b, j+1);
        }
```

A) returns true if `a` and `b` are equal in size, false otherwise
B) returns true if `a` is larger than `b`, false otherwise
C) returns true if `b` is larger than `a`, false otherwise
D) returns true if `a` and `b` have no elements
E) returns the length of array `a + ` length of array `b`

Answer: B

14) Why is the following method one which has infinite recursion?

```
        public int infiniteRecursion(int n)
        {
              if (n > 0) return infiniteRecursion(n) + 1;
              else return 0;
        }
```

A) Because there is no base case
B) Because the base case will never be true
C) Because the recursive call does not move the parameter closer to the base case
D) Because the recursive call moves the problem further away from the base case
E) None of the above, there is no infinite recursion in this method

Answer: C

For the questions below, recall the Towers of Hanoi recursive solution.

15) If there are 2 disks to move from one Tower to another, how many disk movements would it take to solve the problem using the recursive solution?
A) 0
B) 1
C) 2
D) 3
E) 4

Answer: D

16) If there are 6 disks to move from one Tower to another, how many disk movements would it take to solve the problem using the recursive solution?
A) 6
B) 13
C) 31
D) 63
E) 127

Answer: D

17) The solution to the Towers of Hanoi has a(`n`) ________ complexity.
A) `linear`
B) `polynomial`
C) `logarithmic`
D) `exponential`
E) `bad`

Answer: D

18) Which of the following methods would properly compute the value of `x % y` (`x` mod `y`) recursively, assuming that `x` and `y` not negative numbers?
A)
```
public int recursiveMod(int x, int y)
{
       if (x < y) return y;
       else return recursiveMod(x, y - x);
}
```
B)
```
public int recursiveMod(int x, int y)
{
       if (x == y) return 0;
       else return recursiveMod(x - y, y) + 1;
}
```
C)
```
public int recursiveMod(int x, int y)
{
       if (x < y) return x;
       else return recursiveMod(x - y, y) + 1;
}
```
D)
```
public int recursiveMod(int x, int y)
{
       if (x < y) return x;
       else return recursiveMod(x - y, y);
}
```
E)
```
public int recursiveMod(int x, int y)
{
       while (x > y)
             x = x - y;
       return x;
}
```

Answer: D
Explanation: D) The value x % y can be computed by continually subtracting y from x until x < y. Then, the result is x. For instance, 7 % 3 = 4 % 3 = 1 % 3 = 1. The code in answer A - C do not compute x mod y and answer E computes mod but uses an iterative solution, not a recursive one.

19) Define the magnitude of a number as the location of the decimal point from the left of the number (that is, if a number has 4 digits followed by the decimal point, it will have a magnitude of 4). 100 would then have a magnitude of 3 and 55,555.555 would have a magnitude of 5. A partial recursive method is given below to compute a positive int parameter's magnitude. Which answer below is needed to complete the method?
```
       public int magnitude(double x)
       {
          if (x < 1) return 0;
             else return _______;
       }
```
A) `magnitude(x - 1) + 10;`
B) `magnitude(x - 10) + 1;`

C) `magnitude(x / 10) + 10;`
D) `magnitude(x / 10) + 1;`
E) `magnitude(x / 10) * 2;`

Answer: D
Explanation: D) The method must count the number of digits to the left of the decimal point, so it continually divides the value by 10 until the value has no digits to the left of the decimal point (this is the case if value < 1). Each time the value is divided by 10, 1 is added to the return value. 55,555.555 will be divided by 10 five times before it becomes < 1, and thus return 5.

The following method recognizes whether a String parameter consists of a specific pattern and returns true if the String has that pattern, false otherwise. Use this recursive method to answer the questions below.

```
        public boolean patternRecognizer(String a)
        {
          if (a == null) return false;
          else if (a.length( ) = = 1 | | (a.length( ) = = 2 && a.charAt(0) = =
a.charAt(1) ) ) return true;
          else if (a.length( ) = = 2 && a.charAt(0) != a.charAt(1) ) return
false;
          else if (a.charAt(0) == a.charAt(a.length( ) - 1))
               return patternRecognizer(a.substring(1, a.length( ) - 1));
          else return false;
        }
```

26) Which String below would result in `patternRecognizer` returning true?
A) "`abcba`"
B) "`aaabbb`"
C) "`abcde`"
D) "`aabba`"
E) all of the above Strings will result in the method returning true

Answer: A
Explanation: A) The method patternRecognizer returns true if the String is a palindrome. This can be seen by analyzing the code. If the String has 1 character, or 2 characters which are equal, it returns true, otherwise if the first and last characters are the same, it recursively calls itself with the substring starting at character 1 and going to the second to last character (so, since in "abcba" the first and last characters are the same, the method is called recursively with the substring "bcb"). Only if the first and last characters do not match is false returned. The only palindrome in the list of options is a, "abcba".

27) If the method is called as `patternRecognizer(x)` where `x` is "`aa`", what will the result be?
A) `true`
B) `false`
C) a `NullPointerException`
D) a `run-time` error
E) infinite recursion

Answer: A
Explanation: A) One of the base cases tests to see if the String has 2 characters and if so, returns true if the two characters are the same.

28) If the statement `a.substring(1, a.length( ) - 1)` were changed to be (`a.substring(1, a.length( )`), then the method would

A) have infinite recursion unless the String were null
B) always return true
C) always return false
D) return true only if all characters of the String were the same
E) return true only the String had only two characters and they were the same
Answer: D
Explanation: D) The original method recursively calls itself with a substring equal to the String minus the first and last characters. This new version would recursively call itself with a substring equal to the String minus the first character. The method would return true if the first character of the String (or substring) equaled the last character. Since the substring for each recursive call is the same as the previous String less the first character, it winds up comparing each character in the String to the last in the String and returning true only if each character of the String equals the last character. This means all of the characters are the same.

29) A recursive algorithm is superior to an iterative algorithm along which of the following criteria?
A) The recursive algorithm is easier to debug
B) The recursive algorithm is computationally more efficient
C) The recursive algorithm is more elegant
D) The recursive algorithm requires less memory to execute
E) all of the above

Answer: C
Explanation: C) When comparing most recursive algorithms to their iterative counterparts, it is almost always the case that the recursive algorithm is shorter, so it tackles the same problem with less code. This means it is more elegant (which does not necessarily mean it is easier to understand).

30) Aside from writing recursive methods, another way that recursion is often used is to define
A) words in English
B) mathematical functions
C) rules and laws
D) child classes of a parent class in object-oriented programming
E) recursion is used in all of the above
Answer: B
Explanation: B) Mathematics often defines functions recursively for simplicity.

34) The difference between direct and indirect recursion is
A) direct recursion occurs when a method invokes itself; indirect recursion occurs when there is an intervening method
B) indirect recursion occurs when a method invokes itself; direct recursion occurs when there is an intervening method
C) direct recursion only occurs with methods declared to be private; indirect recursion can occur with methods declared to be private, protected, or public
D) indirect recursion only occurs with methods declared to be private; direct recursion can occur with methods declared to be private, protected, or public
E) none of the above

Answer: A
Explanation: A) Direct recursion means that a method directly invokes itself with no intervening method. Indirect recursion occurs when there are one or more intervening methods before the original method is invoked again.

35) An infinite loop and an infinite recursion
A) are different because it is impossible to detect the latter, while it's quite easy to detect the former
B) both continue to repeat indefinitely
C) both will be caught by the compiler
D) both will be caught by the Java Virtual Machine during execution
E) none of the above

Answer: B
Explanation: B) Both infinite loops and recursion are similar in that they continue to repeat indefinitely. Neither can be caught by the compiler or by the runtime (the JVM).

2 True/False Questions

1) A recursive method without a base case leads to infinite recursion.

Answer: TRUE
Explanation: Without the base case, the recursive method calls itself without the ability to stop, and thus leads to infinite recursion.

2) The following method lacks a base case.

```
    public int noBaseCase(int x)
    {
            if (x > 0)
                return noBaseCase(x - 1) + 1;
            else return noBaseCase(x - 2) + 2;
}
```

Answer: TRUE
Explanation: A base case is a condition that, when taken, stops the recursion by not calling the method recursively. In the above method, there is a condition, but whether the condition is true or false, the method invokes itself recursively.

3) Traversing a maze is much easier to do iteratively than recursively.

Answer: FALSE
Explanation: One reason that recursion is appealing is that it automatically backtracks to the point of the last decision. In traversing a maze, when one reaches a dead end, the best place to continue is at the point of the previous intersection (or decision). So, backtracking is performed. To solve a maze iteratively requires implementing a backtracking mechanisms, which is available automatically in recursion.

4) Some problems are easier to solve recursively than iteratively.

Answer: TRUE
Explanation: Since recursion performs backtracking automatically, any problem that requires backtracking is easier to solve using recursion. Such problems include traversing a maze and searching through a "search space", a topic covered in Artificial Intelligence and Advanced Algorithms courses. In some cases, it is easy to find a recursive solution and so the recursive solution is easier than the iterative solution.

5) The following two methods will both compute the same thing when invoked with the same value of x. That is, method1(x) = = method2(x).

```
        public int method1(int x)
        {
                if (x > 0) return method1(x - 1) + 1;
                else return 0;
        }

        public int method2(int x)
        {
                if (x > 0) return 1 + method2(x - 1);
        else return 0;
        }
```

Answer: TRUE
Explanation: The only difference between the two methods is the order of the recursive call and adding 1 to the return value. The difference is denoted by calling the first method head recursive and the second tail recursive. So both methods will compute the same value, but arrive at the value in different ways.

6) Consider the following recursive sum method:

```
public int sum(int x)
{
        if (x = = 0) return 0;
        else return sum(x - 1) + 1;
}
```

If the base case is replaced with "if (x = = 1) return 1;" the method will still compute the same thing.

Answer: FALSE
Explanation: The original method causes infinite recursion if called with a parameter < 0, but works properly if called with any parameter >= 0. With the new base case, the method now works properly if called with any parameter >= 1 but causes infinite recursion if the parameter < 1. So, sum(0) now differs from what it had previously.

7) The recursive method to solve the Towers of Hanoi is usable only if the parameter for the number of disks is 7 or smaller.

Answer: FALSE
Explanation: The Towers of Hanoi solution can be used for any sized disk as long as it is at least 1.

8) A Koch snowflake of order = 1 can be drawn without recursion.

Answer: TRUE
Explanation: The Koch snowflake of order 1 is made up of straight lines. It is not until the order is greater than 1 that recursion is applied.

9) Since iterative solutions often use loop variables and recursive solutions do not, the recursive solution is usually more memory efficient (uses less memory) than the equivalent iterative solution.

Answer: FALSE
Explanation: The recursive solution may use no local variables whatsoever, but the recursive solution usually uses parameters. Every time the method is called recursively, new parameters are required, and so recursively solutions usually use a substantially larger amount of memory than an iterative solution (it depends on how many times the method is called recursively).

10) We can define a list of int values recursively as: a list_item, followed by a comma, followed by a list where a list_item is any int value.

Answer: FALSE
Explanation: The recursive definition does not include a base case so that all lists of int values will be infinitely long!

11) It always is possible to replace a recursion by an iteration and vice versa.

Answer: TRUE
Explanation: Both recursion and iteration are forms of repetition. Whether one phrases the repetition using recursion or iteration is a matter of style, taste, sometimes efficiency, sometimes convenience. But they are equivalent in terms of computation–each may be replaced by the other.

12) The following method correctly adds two ints, returning their sum:

```
public int add(int a, int b)
{
        return (b > 0) ? add(a+1, b-1) : a;
    }
```

Answer: FALSE
Explanation: The add method works fine for ints which are greater than or equal to zero; but it fails if b is less than zero.

13) The following method correctly multiplies two ints so long as both are non-negative:

```
    public int mpy(int a, int b)
    {
        return (b > 0) ? a + mpy(a, b-1) : 0;
    }
```

Answer: TRUE
Explanation: Multiplication is just repeated addition. This method just repeated add a's, b times. So long as both a and b are non-negative the method works fine.

14) If one were to create a Towers of Hanoi puzzle with four towers instead of three, with rules changed appropriately, it should be possible to create a recursive solution for the puzzle where one of the constraints is that at some point in the solution all of the disks must reside on each of the four towers.

Answer: TRUE
Explanation: The solution to this version of the game can be phrased as follows: Use the 3-peg Towers solution to move all the disks from peg A to peg B; then use the 3-peg Towers solution to move all the disks from peg B to peg C; repeat, moving from peg C to peg D.

15) The Koch snowflake has an infinitely long perimeter, but it contains a finite area.

Answer: TRUE
Explanation: As the order of the Koch snowflake increases the perimeter grows exponentially large. In the limit, the perimeter is infinitely long. But, since the entire snowflake can be contained within a bounding square, the area of the snowflake is finite.

3 Free-Form Questions

1) Provide a definition for the terms as they relate to programming: recursion, indirect recursion and infinite recursion.
Answer: Recursion: writing a method that calls itself. Indirect recursion: writing a method that calls other methods that call the original method so that the recursion is not done directly through the first method. Infinite recursion: recursion where the recursion is never stopped by a base case because either the base case was not defined, or the base case is never reached.

3) Rewrite the following iterative method as a recursive method that computes the same thing. NOTE: your recursive method will require an extra parameter.

```
public String reversal(String x)
{
   int y = x.length( );
   String s = "";
   for (int j = y-1; j >=0; j--)
      s += x.charAt(j);
   return s;
}
```

```
Answer:     public String reversal(String x, int j)
            {
                  if (j < 0) return "";
                  else return x.charAt(j) + reversal(x, j-1);
            }
```
where the recursive reversal is called with j = x.length( ) - 1.

4) Rewrite the following iterative method as a recursive method that returns the same String.

```
      public String listOfNumbers( )
      {
         String s = "";
         for (j = 1; j<10; j++)
            s += j;
         return s;
      }
```

```
Answer:     public String listOfNumbers(int j)
            {
                  if (j == 10) return "";
                  else return ""+ j  + listOfNumbers(j+1);
            }
```
where the recursive listOfNumbers is called with j = 1.

9) Assume a function `g(x)` is defined as follows where x is an `int` parameter:

g(x) = g(x - 1) * g (x - 3) if x is even and x > 3
    = g(x - 2) if x is odd and x > 3
    = x otherwise

Write a recursive method to compute g.

```
Answer:  public int g(int x)
{
      if (x > 3 && x % 2 == 0) return g(x - 1) * g (x - 3);
      else if (x > 3) return g(x - 2);
            else return x;
}
```

10) Demonstrate how factorial(4) is computed given the following recursive method for factorial:

```
      public int factorial(int n)
      {
            if (n > 1) return factorial(n - 1) * n;
            else return 1;
      }
```

```
Answer:  factorial(4) = factorial(3) * 4
        factorial(3) = factorial(2) * 3
                factorial(2) = factorial(1) * 2
                        factorial(1) = 1
                factorial(2) = 1 * 2 = 2
        factorial(3) = 2 * 3 = 6
factorial(4) = 6 * 4 = 24
```

18) Explain what a "base case" is in a recursive method.

Answer: The part of a recursive method that does not call itself (either directly or indirectly) to calculate the answer is called the base case. This is the case that "stops" the method, stopping otherwise infinite recursion. Every well written recursive method will have at least one base case.