Multiple-Choice Questions on Recursion

1. Which of the following statements about recursion are true?

   I. Every recursive algorithm can be written iteratively.
   II. Tail recursion is always used in "divide-and-conquer" algorithms.
   III. In a recursive definition, a process is defined in terms of a simpler case of itself.

   (A) I only
   (B) III only
   (C) I and II only
   (D) I and III only
   (E) II and III only

2. Which of the following, when used as the `/* body */` of method `sum`, will enable that method to compute $1 + 2 + \cdots + n$ correctly for any $n > 0$?

```
/** Returns 1 + 2 + ... + n.
 *  Precondition: n is a positive integer.
 */
public int sum(int n)
{
   /* body */
}
```

I.
```
return n + sum(n - 1);
```

II.
```
if (n == 1)
    return 1;
else
    return n + sum(n - 1);
```

III.
```
if (n == 1)
    return 1;
else
    return sum(n) + sum(n - 1);
```

   (A) I only
   (B) II only
   (C) III only
   (D) I and II only
   (E) I, II, and III

3. Refer to the method `stringRecur`.

```
public void stringRecur(String s)
{
    if (s.length() < 15)
        System.out.println(s);
    stringRecur(s + "*");
}
```

When will method `stringRecur` terminate without error?

(A) Only when the length of the input string is less than 15
(B) Only when the length of the input string is greater than or equal to 15
(C) Only when an empty string is input
(D) For all string inputs
(E) For no string inputs

4. Refer to method `strRecur`.

```
public void strRecur(String s)
{
    if (s.length() < 15)
    {
        System.out.println(s);
        strRecur(s + "*");
    }
}
```

When will method `strRecur` terminate without error?

(A) Only when the length of the input string is less than 15
(B) Only when the length of the input string is greater than or equal to 15
(C) Only when an empty string is input
(D) For all string inputs
(E) For no string inputs

Questions 5 and 6 refer to method `result`.

```
public int result(int n)
{
    if (n == 1)
        return 2;
    else
        return 2 * result(n - 1);
}
```

5. What value does `result(5)` return?

(A) 64
(B) 32
(C) 16
(D) 8
(E) 2

6. If $n > 0$, how many times will `result` be called to evaluate `result(n)` (including the initial call)?
   (A) 2
   (B) 2^n
   (C) n
   (D) $2n$
   (E) n^2

7. Refer to method `mystery`.

```
public int mystery(int n, int a, int d)
{
    if (n == 1)
        return a;
    else
        return d + mystery(n - 1, a, d);
}
```

What value is returned by the call `mystery(3, 2, 6)`?
   (A) 20
   (B) 14
   (C) 10
   (D) 8
   (E) 2

8. Refer to method `f`.

```
public int f(int k, int n)
{
    if (n == k)
        return k;
    else
        if (n > k)
            return f(k, n - k);
        else
            return f(k - n, n);
}
```

What value is returned by the call `f(6, 8)`?
   (A) 8
   (B) 4
   (C) 3
   (D) 2
   (E) 1

9. What does method `recur` do?

```
/** x is an array of n integers.
 *  n is a positive integer.
 */
public int recur(int[] x, int n)
{
    int t;
    if (n == 1)
        return x[0];
    else
    {
        t = recur(x, n - 1);
        if (x[n-1] > t)
            return x[n-1];
        else
            return t;
    }
}
```

(A) It finds the largest value in `x` and leaves `x` unchanged.
(B) It finds the smallest value in `x` and leaves `x` unchanged.
(C) It sorts `x` in ascending order and returns the largest value in `x`.
(D) It sorts `x` in descending order and returns the largest value in `x`.
(E) It returns `x[0]` or `x[n-1]`, whichever is larger.

10. Which best describes what the `printString` method below does?

```
public void printString(String s)
{
    if (s.length() > 0)
    {
        printString(s.substring(1));
        System.out.print(s.substring(0, 1));
    }
}
```

(A) It prints string `s`.
(B) It prints string `s` in reverse order.
(C) It prints only the first character of string `s`.
(D) It prints only the first two characters of string `s`.
(E) It prints only the last character of string `s`.

11. Refer to the method `power`.

```
/** Returns base raised to the expo power.
 *  Precondition:
 *   - base is a nonzero real number.
 *   - expo is an integer.
 */
public double power(double base, int expo)
{
    if (expo == 0)
        return 1;
    else if (expo > 0)
        return base * power(base, expo - 1);
    else
        return /* code */;
}
```

Which `/* code */` correctly completes method `power`?
(Recall that $a^{-n} = 1/a^n$, $a \neq 0$; for example, $2^{-3} = 1/2^3 = 1/8$.)

(A) `(1 / base) * power(base, expo + 1)`
(B) `(1 / base) * power(base, expo - 1)`
(C) `base * power(base, expo + 1)`
(D) `base * power(base, expo - 1)`
(E) `(1 / base) * power(base, expo)`

12. Consider the following method.

```
public void doSomething(int n)
{
    if (n > 0)
    {
        doSomething(n - 1);
        System.out.print(n);
        doSomething(n - 1);
    }
}
```

What would be output following the call `doSomething(3)`?

(A) 3211211
(B) 1121213
(C) 1213121
(D) 1211213
(E) 1123211

13. A user enters several positive integers at the keyboard and terminates the list with a sentinel (-999). A `writeEven` method reads those integers and outputs the even integers only, in the reverse order that they are read. Thus, if the user enters

```
3 5 14 6 1 8 -999
```

the output for the `writeEven` method will be

```
8 6 14
```

Assume that the user enters at least one positive integer and terminates the list with −999. Here is the method.

```
/** Postcondition: All even integers in the list are output in
 *                 reverse order.
 */
public static void writeEven()
{
    int num = ...;    //read user input
    if (num != -999)
    {
        /* code */
    }
}
```

Which `/* code */` satisfies the postcondition of method `writeEven`?

I.
```
if (num % 2 == 0)
    System.out.print(num + " ");
writeEven();
```

II.
```
if (num % 2 == 0)
    writeEven();
System.out.print(num + " ");
```

III.
```
writeEven();
if (num % 2 == 0)
    System.out.print(num + " ");
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II, and III

14. Refer to the following recursive method.

```
public int mystery(int n)
{
    if (n < 0)
        return 2;
    else
        return mystery(n - 1) + mystery(n - 3);
}
```

What value is returned by the call `mystery(3)`?

(A) 12
(B) 10
(C) 8
(D) 6
(E) 4

Questions 15 and 16 refer to method `t`.

```
/** Precondition: n is a positive integer. */
public int t(int n)
{
    if (n == 1 || n == 2)
        return 2 * n;
    else
        return t(n - 1) - t(n - 2);
}
```

15. What will be returned by `t(5)`?
    (A) 4
    (B) 2
    (C) 0
    (D) −2
    (E) −4

16. For the method call `t(6)`, how many calls to `t` will be made, including the original call?
    (A) 6
    (B) 7
    (C) 11
    (D) 15
    (E) 25

17. This question refers to methods `f1` and `f2` that are in the same class.

```
public int f1(int a, int b)
{
    if (a == b)
        return b;
    else
        return a + f2(a - 1, b);
}

public int f2(int p, int q)
{
    if (p < q)
        return p + q;
    else
        return p + f1(p - 2, q);
}
```

What value will be returned by a call to `f1(5, 3)`?
    (A) 5
    (B) 6
    (C) 7
    (D) 12
    (E) 15

18. Consider method `foo`.

```
public int foo(int x)
{
    if (x == 1 || x == 3)
        return x;
    else
        return x * foo(x - 1);
}
```

Assuming no possibility of integer overflow, what will be the value of `z` after execution of the following statement? Note that $n! = (n)(n-1)(n-2)\ldots(2)(1)$.

```
int z = foo(foo(3) + foo(4));
```

(A) (15!)/(2!)
(B) 3! + 4!
(C) (7!)!
(D) (3! + 4!)!
(E) 15

Questions 19 and 20 refer to the `IntFormatter` class below.

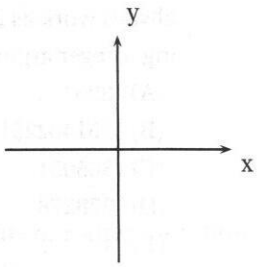
```
public class IntFormatter
{
    /** Write 3 digits adjacent to each other.
     *  Precondition: n is a nonnegative integer.
     */
    public static void writeThreeDigits(int n)
    {
        System.out.print(n / 100);
        System.out.print((n / 10) % 10);
        System.out.print(n % 10);
    }

    /** Insert commas in n, every 3 digits starting at the right.
     *  Precondition: n is a nonnegative integer.
     */
    public static void writeWithCommas(int n)
    {
        if (n < 1000)
            System.out.print(n);
        else
        {
            writeThreeDigits(n % 1000);
            System.out.print(",");
            writeWithCommas(n / 1000);
        }
    }
}
```

19. The method `writeWithCommas` is supposed to print its nonnegative `int` argument with commas properly inserted (every three digits, starting at the right). For example, the integer 27048621 should be printed as 27,048,621. Method `writeWithCommas` does not always work as intended, however. Assuming no integer overflow, which of the following integer arguments will not be printed correctly?
    (A) 896
    (B) 251462251
    (C) 365051
    (D) 278278
    (E) 4

20. Which change in the code of the given methods will cause method `writeWithCommas` to work as intended?
    (A) Interchange the lines `System.out.print(n / 100)` and `System.out.print(n % 10)` in method `writeThreeDigits`.
    (B) Interchange the lines `writeThreeDigits(n % 1000)` and `writeWithCommas(n / 1000)` in method `writeWithCommas`.
    (C) Change the test in `writeWithCommas` to `if (n > 1000)`.
    (D) In the method `writeWithCommas`, change the line `writeThreeDigits(n % 1000)` to `writeThreeDigits(n / 1000)`.
    (E) In the method `writeWithCommas`, change the recursive call `writeWithCommas(n / 1000)` to `writeWithCommas(n % 1000)`.

21. Consider the following method.

```
public static void sketch(int x1, int y1, int x2, int y2, int n)
{
    if (n <= 0)
        drawLine(x1, y1, x2, y2);
    else
    {
        int xm = (x1 + x2 + y1 - y2) / 2;
        int ym = (y1 + y2 + x2 - x1) / 2;
        sketch(x1, y1, xm, ym, n - 1);
        sketch(xm, ym, x2, y2, n - 1);
    }
}
```

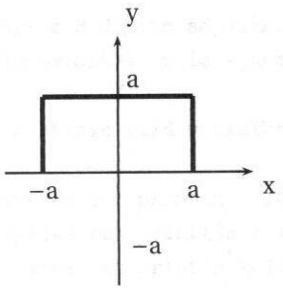


Assume that the screen looks like a Cartesian coordinate system with the origin at the center, and that `drawLine` connects `(x1,y1)` to `(x2,y2)`. Assume also that `x1`, `y1`, `x2`, and `y2` are never too large or too small to cause errors. Which picture best represents the sketch drawn by the method call

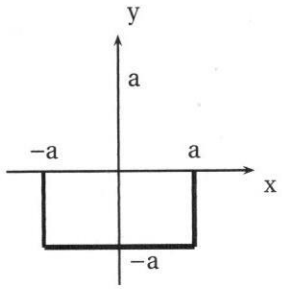
```
sketch(a, 0, -a, 0, 2)
```

where `a` is a positive integer?

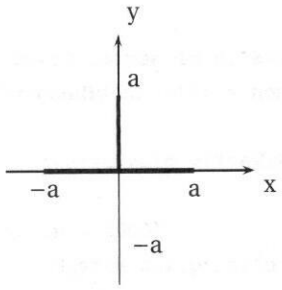
(A)



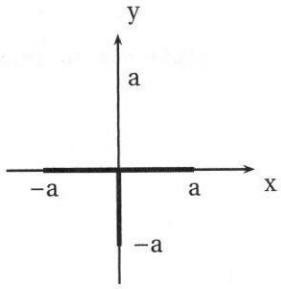
(B)



(C)



(D)



(E)