In all these problems, we'll assume that the following has been done in the appropriate places:

```
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
```

1. What is the exact output of the program below. Indicate a blank line in the output by writing `blank line` .

```
public static void main(String[] args)
{
    int n = 4, k = 2;

    System.out.println(++n );
    System.out.println( n );

    System.out.println( n++ );
    System.out.println( n );

    System.out.println( -n );
    System.out.println( n );                    ...

    System.out.println( --n );
    System.out.println( n );

    System.out.println( n-- );              ...
    System.out.println( n );

    System.out.println( n + k );                ...
    System.out.println( n );
    System.out.println( k );                    ...
    System.out.println("" + n + " " + k );      ...

    System.out.println( n );                    ...
    System.out.println( "  " + n );             ...
    System.out.println( " n" );                 ...
    System.out.println( "\n" );                 ...

    System.out.println( " n * n = "); //CAREFUL!
    System.out.println( n * n );                ...
    System.out.println( 'n' );              ...
}
```

2. What is the output of the program below?

```
public static void main(String[] args)
{
    int n = 3;
    while (n >= 0)
    {
        System.out.println( n * n );
        --n;
    }

    System.out.println( n );

    while (n < 4)
        System.out.println( ++n );

    System.out.println( n );

    while (n >= 0)
        System.out.println( (n /= 2) );

}
```

3. What is the output of the program below?

```
public static void main(String[] args)
{
    int n;

    System.out.println( (n = 4) );
    System.out.println( (n == 4) );
    System.out.println( (n > 3) );
    System.out.println( (n < 4) );
    System.out.println( (n = 0) );
    System.out.println( (n == 0) );
    System.out.println( (n > 0) );
    System.out.println( (n == 0 && true) );
    System.out.println( (n == 0 || true) );
    System.out.println( (!(n == 2) ));

}
```

4. What is the output of the following program?

```
enum colorType {red, orange, yellow, green, blue, violet};

public static void main(String[] args)
{
    colorType shirt, pants;

    shirt = colorType.red;
    pants = colorType.blue;

    System.out.println("  " + shirt + "  " + pants );
}
```

5. What is the output when the following code fragment is executed?

```
int i = 5, j = 6, k = 7, n = 3;
System.out.println( i + j * k - k % n );
System.out.println( i / n );
```

6. What is the output when the following code fragment is executed?

```
int found = 0, count = 5;
if (!found || --count == 0)
    System.out.println( "danger" );
System.out.println( "count = " + count );
```

7. What is the output when the following code fragment is executed?

```
char [] title = {'T', 'i', 't', 'a', 'n', 'i', 'c'};

ch = title[1];
title[3] = ch;

System.out.println( title );
System.out.println( ch );
```

8. Suppose that the following code fragment is executed.

```
String message;

System.out.println( "Enter a sentence on the line below." );
message = scan.next();

System.out.println( message );
```

a. Suppose that in response to the prompt, the interactive user types the following line and presses Enter:

```
Please go away.
```

What will the output of the code fragment look like?

b. Suppose that we modify the program and use `scan.nextLine()` instead of `scan.next()`, and that in response to the prompt, the interactive user types the following line and presses Enter:

```
Please stop bothering me.
```

What will the output of the code fragment look like?

9. Write a for loop that outputs all the *even* numbers between 0 and 20 in reverse order, from the largest to the smallest.

10. Write a for loop that outputs all the *odd* numbers between 0 and 20 in reverse order, from the largest to the smallest.

11. The nested conditional statement shown below has been written by an inexperienced Java programmer. The behavior of the statement is not correctly represented by the formatting.

```
if (n < 10)
    if (n > 0)
        System.out.println("The number is positive.");
else
    System.out.println("The number is _____________.");
```

a. What is the output of the statement if the variable n has the value 7 ? If n has the value 15 ? If n has the value -3 ?

b. Correct the syntax of the statement so that the logic of the corrected statement corresponds to the formatting of the original statement. Also, replace the blank with an appropriate word or phrase.

c. Correct the formatting of the (original) statement so that the new format reflects the logical behavior of the original statement. Also, replace the blank with an appropriate word or phrase.

12. The loop shown below has been written by an inexperienced Java programmer. The behavior of the loop is not correctly represented by the formatting.

```
int n = 10;

while (n > 0)
    n /= 2;
    System.out.println(n * n);
```

a. What is the output of the loop as it is written?
b. Correct the syntax of the loop so that the logic of the corrected loop corresponds to the formatting of the original loop. What is the output of the corrected loop?
c. Correct the formatting of the (original) loop so that the new format reflects the logical behavior of the original loop.

13. Remove all the unnecessary tests from the nested conditional statement below.

```
float income;

System.out.println("Enter your monthly income: ";
income = scan.nextFloat();

if (income < 0.0)
    System.out.println("You are going farther into debt every month.");
else if (income >= 0.0  &&  income < 1200.00)
    System.out.println("You are living below the poverty line.");
else if (income >= 1200.00  &&  income < 2500.00)
    System.out.println("You are living in moderate comfort.");
else if (income >= 2500.00)
    System.out.println("You are well off.");
```

14. Answer the questions below concerning the following fragment of code.

```
int n;
System.out.println("Enter an integer: ";
n = scan.nextInt();

if (n < 10)
    System.out.println("less than 10");

else if (n > 5)
    System.out.println("greater than 5");

else
```

```
    System.out.println("not interesting");
```

a. What will be the output of the fragment above if the interactive user enters the integer value 0 ?
b. What will be the output of the fragment above if the interactive user enters the integer value 15 ?
c. What will be the output of the fragment above if the interactive user enters the integer value 7 ?
d. What values for n will cause the output of the fragment above to be "not interesting"?

15. Rewrite the following code fragment so that it uses a "do - while" loop to accomplish the same task.

```
int n;

System.out.println("Enter a non-negative integer: ";
n = scan.nextInt();

while (n < 0)
{
    System.out.println("The integer you entered is negative.");
    System.out.println("Enter a non-negative integer: ";
    n = scan.nextInt();
}
```

16. In the code fragment below, the programmer has almost certainly made an error in the first line of the conditional statement.
a. What is the output of this code fragment as it is written?

```
int n = 5;
if (n == 0)
    System.out.println("n is zero.");
else
    System.out.println("n is not zero.");

    System.out.println("The square of n is " + n * n + ".");
```

17. What is the output when the following code fragment is executed?

```
int n, k = 5;
n = (100 % k == 0 ? k + 1 : k - 1);
System.out.println("n = " + n + "   k = " + k);
```

18. What is the output when the following code fragment is executed?

```
int   n;
float x = 3.8;
```

```
n = (int) x;
System.out.println("n = " + n);
```

19. What is the output when the following code fragment is executed? Rewrite the fragment to obtain an equivalent code fragment in which the body of the loop is a simple statement instead of a compound statement.

```
int i = 5;
while (i > 0)
{
    --i;
    System.out.println(i);
}
```

20. The following loop is an endless loop: when executed it will never terminate. What modification can be made in the code to produce the desired output?

```
System.out.println("Here's a list of the ASCII values of all the upper"
                   + " case letters.");
char letter = 'A';
while (letter <= 'Z')
    System.out.println(letter + "  " + (int)letter);
```

21. Write a function named "sumFromTo" that takes two integer arguments, call them "first" and "last", and returns as its value the sum of all the integers between first and last inclusive. Thus, for example,

```
System.out.println(sumFromTo(4,7));  // prints  22  because 4+5+6+7 = 22
System.out.println(sumFromTo(-3,1)); // prints  -5  because
                                     // -3+(-2)+(-1)+0+1 = -5
System.out.println(sumFromTo(7,4));  // prints  22  because 7+6+5+4 = 22
System.out.println(sumFromTo(9,9));  // prints  9
```

22. Write a function named "enough" that takes one integer argument, call it "goal" and returns as its value the smallest positive integer n for which 1+2+3+. . . +n is at least equal to goal . Thus, for example,

```
System.out.println(enough(9)); // will print  4  because  1+2+3+4 ≥ 9
                               // but 1+2+3<9
System.out.println(enough(21));// will print  6  because 1+2+ ...+6 ≥ 21
                               // but 1+2+ . . . 5<21
System.out.println(enough(-7));// will print  1  because  1 ≥ -7 and 1 is
                               // the smallest positive integer
System.out.println(enough(1)); // will print  1  because  1 ≥ 1 and 1 is
                               // the smallest positive integer
```

DEFINITION: A positive integer d is called a divisor of an integer n if and only if the remainder after n is divided by d is zero. In this case we also say that " d divides n ", or that " n is divisible by d ". Here are some examples:

- 7 is a divisor of 35 ; that is, 35 is divisible by 7 .
- 7 is a not a divisor of 27 ; that is, 27 is not divisible by 7 .
- 1 is a divisor of 19 ; that is, 19 is divisible by 1 (in fact, 1 is a divisor of every integer n).
- 12 is a divisor of 0 ; that is, 0 is divisible by 12 .

In Java one can test the expression `n % d` to determine whether d is a divisor of n .

The greatest common divisor of a pair of integers m and n (not both zero) is the largest positive integer d that is a divisor of both m and n . We sometimes use the abbreviation "g.c.d." for "greatest common divisor. Here are some examples: 10 is the g.c.d. of 40 and 50 ; 12 is the g.c.d. of 84 and -132 ; 1 is the g.c.d. of 256 and 625 ; 6 is the g.c.d. of 6 and 42 ; 32 is the g.c.d. of 0 and 32 .

23. Write a function named "`gcd`" that takes two positive integer arguments and returns as its value the greatest common divisor of those two integers. If the function is passed an argument that is not positive (i.e., greater than zero), then the function should return the value 0 as a sentinel value to indicate that an error occurred. Thus, for example,

```
System.out.println(gcd(40,50));   // will print  10
System.out.println(gcd(256,625)); // will print  1
System.out.println(gcd(42,6));    // will print  6
System.out.println(gcd(0,32));    // will print  0  (even though  32  is
                                  // the g.c.d.)
System.out.println(gcd(10,-6));   // will print  0  (even though  2  is
                                  // the g.c.d.)
```

24. A positive integer n is said to be prime (or, "a prime") if and only if n is greater than 1 and is divisible only by 1 and n . For example, the integers 17 and 29 are prime, but 1 and 38 are not prime. Write a function named "isPrime" that takes a positive integer argument and returns as its value the integer 1 if the argument is prime and returns the integer 0 otherwise. Thus, for example,

```
System.out.println(isPrime(19));   // will print  1
System.out.println(isPrime(1));    // will print  0
System.out.println(isPrime(51));   // will print  0
System.out.println(isPrime(-13));  // will print  0
```

25. Write a function named "`digitName`" that takes an integer argument in the range from 1 to 9 , inclusive, and prints the English name for that integer on the computer screen. No newline character should be sent to the screen following the digit name. The function should not return a value. The cursor should remain on the same line as the name that has been printed. If the argument is not in the required

range, then the function should print "digit error" without the quotation marks but followed by the newline character. Thus, for example,

the statement	`digitName(7);`	should print `seven` on the screen;
the statement	`digitName(0);`	should print `digit error` on the screen and place the cursor at the beginning of the next line.

26. Write a function named "`reduce`" that take one argument that is an array containing two positive integer values, and treats them as the numerator and denominator of a fraction, and reduces the fraction. That is to say, each of the two elements will be modified by dividing it by the greatest common divisor of the two integers. The function should return the value `false` (to indicate failure to reduce) if either of the two arguments is zero or negative, and should return the value `true` otherwise. Thus, for example, if `m` and `n` have been declared to be integer variables in a program, then

```
int[] fraction = {25, 15};
if (reduce(fraction))
    System.out.println("" + fraction[0] + '/' + fraction[1]);
else
    System.out.println("fraction error");
```

will produce the following output:

```
5/3
```

Note that the values of the fraction were modified by the function call. Similarly,

```
int[] fraction = {15, 50};
if (reduce(fraction))
    System.out.println("" + fraction[0] + '/' + fraction[1]);
       else
System.out.println("fraction error");
```

will produce the following output:

```
3/10
```

Here is another example.

```
int[] fraction = {25, 0};
if (reduce(fraction))
    System.out.println("" + fraction[0] + '/' + fraction[1]);
else
    System.out.println("fraction error");
```

will produce the following output:

```
fraction error
```

The function `reduce` is allowed to make calls to other functions that you have written.