Section I. Multiple Choice Questions (MCQs)

Each MCQ has one correct answer. There is no penalty for wrong answers.

1. What is the value of variable **y** after the execution of the following statement?

```
double y = 5/2 + 7%3 + 8;
```

A. **11.5**
B. **11.0**
C. **10.5**
D. **10.0**
E. **0.0**

2. What is printed by the following program fragment?

```
int a = (int)Math.ceil(2.6);
int b = (int)Math.ceil(-2.6);
System.out.println("a = " + a + "; b = " + b);
```

A. **a = 2; b = -2**
B. **a = 3; b = -3**
C. **a = 2; b = -3**
D. **a = 3; b = -2**
E. None of the above

3. What is printed by the following program?

```
class Q3 {
  public static void main(String[] args) {
    int x = 2;
    System.out.println("Ans = " + what(x) + x);
  }

  public static int what (int x) {
    x = 3;
    return x * x;
  }
}
```

A. **Ans = 93**
B. **Ans = 92**
C. **Ans = 42**
D. **Ans = 11**
E. None of the above

4. What is true about the following program?

```
import java.util.*;

class Q4 {

  public static void main(String[] args) {

    int x = 20;
    Scanner input = new Scanner(System.in);
    int n = input.nextInt();

    if (n != 0 && x/n > 10) {
      x++;
    } else {
      x /= 2;
    }
    System.out.println(x);

  }
}
```

A. A syntax error will be reported during compilation.
B. An **`ArithmeticException`** will be thrown if user input is 0.
C. The program will print out 21 if user input is 0.
D. The program will print out 10 if user input is 2.
E. None of the above.

5. Which statement is true about the following program fragment?

```
int x = 1, y = 1;
while (x < 100) {
  x *= 3;
  y++;
}
```

A. The value of **`x`** after the loop is 103.
B. The value of **`y`** after the loop is 6.
C. Compilation error
D. The **`while`** loop is infinite.
E. None of the above

6. Which of the following Boolean expressions is equivalent to the Boolean expression?

```
!(x < 0 && y < 0)
```

A. `!(x < 0) && !(y < 0)`

B. `!(x >= 0) || !(y >= 0)`

C. `x >= 0 && y >= 0`

D. `x > 0 && y > 0`

E. None of the above

7. What is printed by the following program fragment?

```
int[] list = {0, 1, 2, 3, 4};
int len = list.length;
for (int a = 0; a < len/2; a++) {
  int tmp = list[a];
  list[a] = list[len-a-1];
  list[len-a-1] = tmp;
}
System.out.println(Arrays.toString(list));
```

A. `[0, 0, 0, 0, 0]`

B. `[0, 1, 2, 3, 4]`

C. `[4, 3, 2, 1, 0]`

D. `[4, 0, 2, 1, 2]`

E. None of the above

8. What is printed by the following program fragment?

```
int[] list = {0, 1, 2, 3, 4};

for (int a = 0; a < list.length; a++) {
  for (int b = 0; b < a; b++) {
    list[a] += list[b];
  }
}

System.out.println(Arrays.toString(list));
```

A. `[0, 0, 0, 0, 0]`

B. `[0, 1, 3, 7, 15]`

C. `[0, 1, 3, 6, 10]`

D. `[10, 31, 93, 278, 832]`

E. None of the above

9. Suppose **arr** is an array of distinct integers. Consider the following method.

```
public static boolean mystery(int[] arr){
  for (int i = 0; i < arr.length-1; i++) {
    if (arr[i] < arr[i+1]) {
      return false;
    }
  }
  return true;
}
```

Which of the following statement is true?

A. This method returns **true** if the integers in **arr** are arranged in ascending order, or **false** otherwise.
B. This method returns **false** if the integers in **arr** are arranged in ascending order, or **true** otherwise.
C. This method returns **false** if the integers in **arr** are arranged in descending order, or **true** otherwise.
D. This method returns **true** if the integers in **arr** are arranged in descending order, or **false** otherwise.
E. This method is buggy such that a runtime exception will be thrown.

10. What is printed by the following program fragment?

```
int[] list = new int[5];
for (int i = 0; i < list.length; i++) {
  list[i]++;
  for (int j = i+1; j < list.length; j += 2) {
    list[j] += list[i];
  }
}

System.out.println(Arrays.toString(list));
```

A. `[1, 2, 3, 5, 7]`
B. `[1, 2, 3, 5, 8]`
C. `[2, 2, 6, 6, 18]`
D. `[1, 2, 4, 8, 16]`
E. None of the above

11. In a physical fitness proficiency test, points are awarded depending on the number of sit-ups performed according to this table.

Number of sit-ups	< 10	10 - 19	20 - 39	>= 40
Points awarded	0	1	2	3

Which of the following methods computes the points correctly, assuming **`situp`** is a non-negative integer?

(i)

```
// Pre-condition: situp >= 0
public static int compute_points_1(int situp) {

  int points = 0;

  points += (situp >  9) ? 1 : 0;
  points += (situp > 19) ? 1 : 0;
  points += (situp > 39) ? 1 : 0;

  return points;
}
```

(ii)

```
// Pre-condition: situp >= 0
public static int compute_points_2(int situp) {

  int points = 0;

  if (situp >= 40) {
    points += 3;
  } else if (situp >= 20) {
    points += 2;
  } else {
    points++;
  }

  return points;
}
```

(iii)

```
// Pre-condition: situp >= 0
public static int compute_points_3(int situp) {

  int points = 0;
  if (situp > 39) {
    points++;
  }
  if (situp > 19) {
    points++;
  }
  if (situp > 9) {
    points++;
  }

  return points;
}
```

(iv)

```
// Pre-condition: situp >= 0
public static int compute_points_4(int situp) {

  int points = 0;
  switch (situp/10) {
    case 1:
      points = 1; break;
    case 2: case 3:
      points = 2; break;
    default:
      points = 3;
  }

  return points;
}
```

A. (i) and (iii) only
B. (iii) only
C. (ii) and (iii) only
D. (i), (iii) and (iv) only
E. (i), (ii), (iii) and (iv)

12. Suppose **n** is an integer greater than 1, what does the following method return?

```
// Pre-condition: n > 1
public static boolean confuse(int n){
  int i = 1;
  while (i*i <= n) {
    i++;
    if (n%i == 0) {
      return false;
    }
  }
  return true;
}
```

A. It always returns **false**.
B. It always returns **true**.
C. It always returns **true** if **n** is a prime number, or **false** otherwise.
D. It always returns **true** if **n** is a composite number, or **false** otherwise.
E. None of the above

13. You are given the following **printLine()** method.

```
public static void printLine(int n) {
  while (n > 0) {
    System.out.println("Line " + n);
    n--;
  }
}
```

How many lines will be printed by the following loop?

```
for (int i = 1; i <= k; i++) {
  printLine(i);
}
```

A. **k**
B. **k*k**
C. **k*(k+1)/2**
D. **k*(k-1)/2**
E. None of the above

14. Which of the following statement best describes the method **dothing()**, assuming **x > y > 0**?

```
// Pre-condition: x > y > 0
public static int dothing(int x, int y) {
  int i = 1, m = x;
  while (m%y != 0) {
    i++;
    m = i * x;
  }
  return m;
}
```

A. It returns **x** raised to the power of **y**, i.e. x^y.
B. It returns **y** raised to the power of **x**, i.e. y^x.
C. It returns the greatest common factor of **x** and **y**, i.e. the largest integer divisor of both **x** and **y**.
D. It returns the least common multiple of **x** and **y**, i.e. the smallest integer that has both **x** and **y** as a factor.
E. It returns the smallest common factor of **x** and **y**, i.e. the smallest positive integer divisor of both **x** and **y**.

15. Method **mys()** has the following input-output relationship:

num	2222	1111	5217	6534
mys(num)	0	1111	1011	110

```
public static int mys(int num) {
  int bin = 0, k = 1;
  while (num > 0) {
    int d = num % 10;
    num /= 10;
    bin = XXXXXXXXX;
    k *= 10;
  }
  return bin;
}
```

Which of the following expressions can be substituted into **XXXXXXXXX** to produce the desired output?

A. **bin * k + (d%2)**
B. **bin * k + 1 - (d%2)**
C. **bin + (d%2) * k**
D. **bin + (1-(d%2)) * k**
E. None of the above

16. The **`Math.random()`** method returns a random **`double`** value greater than or equal to 0.0 and less than 1.0. Suppose **`p`** is a positive **`int`** variable, which of the following expressions generates a random **odd** integer in the range [1, **`p`**], both inclusive?

    A. `(int)(Math.random()*p/2) * 2 + 1`

    B. `(int)(Math.random()*(p+1)/2) * 2 + 1`

    C. `(int)Math.random()*(p+1)/2 * 2 + 1`

    D. `(int)(Math.random()*p/2 * 2) + 1`

    E. `(int)(Math.random()*(p+1))/2 * 2 + 1`

17. Given an array containing zeros and ones only, such as **`arr`** = {0, 1, 1, 0, 1, 1, 0}, the following method **`find_idx()`** returns the array index of the **`n`**th occurrence of 1 (**`n`** > 0). It will return **`-1`** if the **`n`**th occurrence of 1 cannot be found. For example, method call **`find_idx(arr, 3)`** will return **`4`**, which is the index of the **`3`**th occurrence of 1; method call **`find_idx(arr, 5)`** will return **`-1`**.

```
public static int find_idx(int[] arr, int nth) {

  int nth_idx = -1;

  for (int i = 0; i < arr.length; i++) {

    /*
     * code to process array
     */

  }

  return nth_idx;
}
```

Which of the following statements can be used in place of the comment to produce the desired output?

(i)

```
if (arr[i] == 1) {
  if (nth == 0) {
    nth_idx = i;
    break;
  } else {
    nth--;
  }
```

```
}
```

(ii)

```
if (arr[i] == 1) {
  nth--;
  if (nth == 0) {
    return i;
  }
}
```

(iii)

```
nth -= arr[i];
if (nth == 0) {
  nth_idx = i;
  break;
}
```

A. (i) and (ii) only
B. (i) and (iii) only
C. (ii) and (iii) only
D. (i), (ii) and (iii) only
E. None of the above

Section II. TRACING QUESTIONS

Write down output of the following program / program fragments.

18.

```
int a = 2, b = 8;

while (a < 4) {
  a++;
  System.out.print(a + b);
  b -= 1;
}
System.out.println(a + b);
```

19.

```
int[] array = {1, 2, 3, 4, 5};
int j = 0;

for (int i = 0; i < 10; i++) {
  System.out.print(array[j]);
  j = (j+2) % array.length;
}
System.out.println();
```

20.

```
int i;
for (i = 1; i < 10; i++) {
  if (i%5 == 0) {
    break;
  }
}
System.out.println(i==10);
```

21.

```
int a0 = 0, a1 = 0, a2 = 1, a3 = 1;
for (int n = 1; n < 5; n++) {
  a0 = a1;
  a1 = a2;
  a2 = a3;
  a3 = a0 + a1 + a2;
}
System.out.println(a3);
```

22.

```
class Q22 {

  public static void main(String[] args) {

    int x = 20, y = 10;
    y += f1(x, y);

    System.out.println(y);
  }

  public static int f1(int x, int y) {

    x++;
    y = f2(x);
    System.out.print(x);

    return x;
  }

  public static int f2(int y) {

    while (y < 100) {
      y *= 2;
    }
    System.out.print(y);

    return y;
  }
}
```

23.

```
int count = 0;
for (int x = 0; x < 10; x += 2) {
  for (int y = 10 - x; y > x; y--) {
    count++;
  }
}
System.out.println(count);
```

24.

```
int[] arr = {4, 2, 3, 0, 1};

int count = 0, value = arr[0];

while (arr[value] > 0) {
  count++;
  arr[value] = arr[arr[value]];
}

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

25.

```
int x = 10, i = 0;

do {
  for (int j = 0; j < 20; j++) {
    if (j%2 == 0) {
      x += j;
    } else {
      x += j-1;
    }
  }
  i++;
} while (i < 5);

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

=== END OF PAPER ===

Suggested answers

1. B
2. D
3. B
4. D
5. B
6. E
7. C
8. B
9. D
10. B
11. A
12. E
13. C
14. D
15. C
16. A
17. C
18. 111110
19. 1352413524
20. false
21. 13
22. 1682131
23. 18
24. 3
25. 910

Explanation to selected questions:

Q6:

The expression `!(x < 0 && y < 0)` can be rewritten as `x >= 0 || y >= 0`.

If you don't know the conversion, you can just substitute with numbers and filter out incorrect options.

Q9:

If any array element is smaller than its right neighbor, the mystery method will immediately terminate and return false.

The method will return true only if every element is bigger than its right neighbor. In that case, the array is in descending order.

Q11:

(ii) is incorrect because if situp is less than 10, point will still increase by 1.

(iv) is also incorrect because if situp is less than 10, point will increase by 3 (default clause).

Q12:

The method can correctly check prime number for most positive integers except 2. For the corner case 2, the method will return false.

So, we cannot say the method always returns true if n is a prime number because it doesn't work for n = 2.

Q16:

This option is the correct answer:

(int)(Math.random()*p/2) * 2 + 1

This is because:

1. **Math.random()*p** gives a random double value in the range [0, p). Note that p is exlcusive.

2. **Math.random()*p/2** gives a random double value in the range [0, p/2)

3. **(int)(Math.random()*p/2)** gives a random integer in the list [0, 1, 2... p/2)

4. **(int)(Math.random()*p/2) * 2** gives a random integer in the list [0, 2, 4... p). Note that p is exclusive.

5. **(int)(Math.random()*p/2) * 2 + 1** gives a random integer in the list [1, 3, 4... p+1). Since p+1 is exclusive, the biggest possible number is p.

So this option generates a random odd integer in [1, p]

This following option is wrong because typecast has a higher priority, so **(int)Math.random()** gives 0 and the whole expression results in 0.

(int)Math.random()*(p+1)/2 * 2 + 1

Q17:

The given array **arr** contains zeros and ones only. The question asks to find out the index of the nth occurrence of ones in the array.

An example is given in the question:

arr = {0, 1, 1, 0, 1, 1, 0},

Method call **find_idx(arr, 3)** will return 4 because the 3rd appearance of one is on array index 4.

Method call **find_idx(arr, 5)** will return -1 because there is no 5th appearance of one in the given array (only 4 ones).

You can go through all three options one by one using above two examples, checking whether respective options give the correct return value. The three options share similar idea (i.e. go through array element by element, decrease **n** by 1 when value one is encountered. When n decrease to zero, we find the position of nth element.)

However, option (i) is a wrong implementation because it actually returns the index of (n+1)th occurrence. You can use the example **find_idx(arr, 3)** to trace and convince it.

Q25:

Consider the inner loop:

when **j** is even, we add 0, 2, 4, ... 18 to **x**

when **j** is odd, we still add 0, 2, 4, ... 18 to **x**

So we actually add 2 * (0 + 2 + 4...18) = 180 to **x**

Outer loop is independent of inner loop and will run 5 times.

So in total we add 180 * 5 = 900 to **x**.

Don't forget that initial value of **x** is 10. So the final value of **x** is 910.