Selection

One-way if Statements: A one-way if statement executes a block of code only if a specified condition is true. It's called "one-way" because there's only one possible path of execution if the condition is true. If the condition is false, the program skips the block of code.

```
if (condition) {
    // code to execute if the condition is true
}
```

Two-way if Statements: A two-way if statement has two paths of execution. It consists of an `if` part and an `else` part. If the condition in the `if` part is true, the first block of code is executed. If the condition is false, the program executes the code in the `else` part.

```
if (condition) {
    // code to execute if the condition is true
} else {
    // code to execute if the condition is false
}
```

Nested if Statements: These are if statements within other if statements. This is useful for checking multiple conditions in a sequence or for handling more complex decision-making scenarios.

```
if (condition1) {
    // code to execute if condition1 is true
    if (condition2) {
        // code to execute if both condition1 and condition2 are true
    }
}
```

Logical Operators: Logical operators are used to form complex conditions by combining two or more conditions. The most common logical operators in Java are `&&` (logical AND), `||` (logical OR), and `!` (logical NOT).

- `&&` returns true if both conditions are true.
- `||` returns true if at least one of the conditions is true.
- `!` is used to reverse the result of a condition.

Example:

```
if (condition1 && condition2) {
    // code to execute if both condition1 and condition2 are true
}
```

The & and | and ^ Operators: In Java, `&` and `|`and `^` are bitwise operators that can also be used as logical operators. When used as logical operators:

- `&` checks both conditions, even if the first one is false.
- `|` checks both conditions, even if the first one is true.
- `^` XOR (exclusive OR), is 1 (true) if the bits being compared are different and 0 (false) if they are the same.

This is different from `&&` and `||`, which perform short-circuit evaluation (they do not evaluate the second condition if the first one is sufficient to determine the result). The `&` and `|` operators are rarely used for logical operations in everyday coding because of their lack of short-circuit evaluation.

Example:

```
if (condition1 & condition2) {
    // code to execute if both condition1 and condition2 are true
}
```

1. What will the following Java code snippet output?

```
class punchcard {
    public static void main(String[] args) {
        int x = 4, y = 9;
        if (x * y < 50)
            x += y;
        else
            y -= x;
        System.out.println("x: " + x + ", y: " + y);
    }
}
```

A) x: 4, y: 9
B) x: 13, y: 9
C) x: 4, y: 5
D) x: 13, y: 5

---

2. Which of these is a correct one-way if statement in Java?

A) `if (x = 5) {...}`
B) `if x > 5 {...}`
C) `if (x > 5) {...}`
D) `if x > 5 then {...}`

---

3. Given the following Java code, what will be the output?

```
class punchcard {
    public static void main(String[] args) {
        int a = 5, b = 5;
        if (++a > 5 && ++b > 5)
            System.out.println("Branch 1");
        else
            System.out.println("Branch 2");
        System.out.println("a: " + a + ", b: " + b);
    }
}
```

A) Branch 1, a: 6, b: 6
B) Branch 1, a: 6, b: 5
C) Branch 2, a: 6, b: 6
D) Branch 2, a: 6, b: 5

---

4. Which of the following is not a valid two-way if statement in Java?

A)

```
if (a > b)
    System.out.println("A");
```

```
    else
        System.out.println("B");
```

B)

```
    if (a > b) {
        System.out.println("A");
    }
    else {
        System.out.println("B");
    }
```

C)

```
    if a > b {
        System.out.println("A");
    }
    else {
        System.out.println("B");
    }
```

D)

```
    if (a > b)
        System.out.println("A");
    else if (a < b)
        System.out.println("B");
    else
        System.out.println("C");
```

---

5. What will be the output of the following Java code?

```
class punchcard {
    public static void main(String[] args) {
        int a = 15, b = 20;
        if (a > 10) {
            if (b > 15) {
                System.out.println("Path 1");
            } else if (a + b > 30) {
                System.out.println("Path 2");
            }
        } else {
            System.out.println("Path 3");
        }
    }
}
```

A) Path 1
B) Path 2
C) Path 3
D) No output

---

6. Which statement about nested if statements is correct?

A) Nested if statements cannot be used in Java.
B) The else block of an outer if statement can contain another if statement.

C) You can only nest if statements up to two levels deep.

---

7. What is the result of the expression `(true && false) || (false && true)`?

A) true
B) false
C) Error

---

8. Consider the following Java code. What is the output?

```
class punchcard {
    public static void main(String[] args) {
        boolean a = false, b = true, c = false;
        if (a || (b && c))
            System.out.println("True path");
        else
            System.out.println("False path");
    }
}
```

A) True path
B) False path
C) No output
D) Error

---

9. Given the following Java code, what is the output?

```
class punchcard {
    public static void main(String[] args) {
        int a = 5, b = 5;
        if (++a > 5 || b++ > 5)
            System.out.println("Branch 1");
        else
            System.out.println("Branch 2");
        System.out.println("a: " + a + ", b: " + b);
    }
}
```

A) Branch 1, a: 6, b: 6
B) Branch 1, a: 6, b: 5
C) Branch 2, a: 6, b: 6
D) Branch 2, a: 6, b: 5

---

10. Consider the following Java code. What is the output?

```
class punchcard {
    public static void main(String[] args) {
        int x = 4, y = 5;
        if ((x & y) > 2)
            System.out.println("Path A");
        else
            System.out.println("Path B");
```

```
    }
}
```

A) Path A
B) Path B
C) No output
D) Error

---

11. Which statement is true about the `&` and `|` operators when used with boolean values?

A) They perform short-circuit evaluation.
B) They always evaluate both operands.
C) They cannot be used with boolean values.
D) none of the above

Answers:

1. B
2. C
3. A
4. C
5. A
6. B
7. B
8. B
9. B
10. A
11. B