SECTION 1: Error-Driven Learning Assignment: Loop Errors

Snippet 1:

```
public class InfiniteForLoop {
public static void main(String[] args) {
for (int i = 0; i < 10; i--) {
System.out.println(i);
 }
 }
 }
```

// Error to investigate: Why does this loop run infinitely? How should the loop control variable be adjusted?

This loop runs infinitely because the i-- is decrementing the value of i before checking it still less than 10, leading to the loop condition always being true.

Correct Code:

```
 public class InfiniteForLoop {
 public static void main(String[] args) {
 for (int i = 0; i < 10; i++) {
 System.out.println(i);
 }
 }
 }
```

Snippet 2:

```
public class IncorrectWhileCondition {
public static void main(String[] args) {
int count = 5;
while (count = 0) {
 System.out.println(count);
count--;
}
}
}
```

// Error to investigate: Why does the loop not execute as expected? What is the issue with the condition in the `while` loop?

Error: class IncorrectWhileCondition is public, should be declared in a file named IncorrectWhileCondition.java

```
public class IncorrectWhileCondition {
       ^
```

Error: incompatible types: int cannot be converted to boolean

```
while (count = 0) {
```

In this assignment operator (=) is used instead of a comparison operator (==),the loop will always execute once because the condition will always be true due to the assignment happening within the loop check.

Correct Code :

```
class IncorrectWhileCondition {
public static void main(String[] args) {
int count = 5;
while (count != 0) {
 System.out.println(count);
count--;
}
}
}
```

Snippet 3:

```
public class DoWhileIncorrectCondition {
public static void main(String[] args) {
int num = 0; do {
System.out.println(num);
num++;
} while (num > 0);
 }
 }
```

// Error to investigate: Why does the loop only execute once? What is wrong with the loop condition in the `do-while` loop?

The loop only executes once because the condition num > 0 is false when the loop starts, because num = 0; do-while loop executes the body at least once, it prints the initial value of num (0) and then exit because the condition becomes false after the increment.

Correct Code:

```
public class DoWhileIncorrectCondition {
public static void main(String[] args) {
int num = 0;
do {
System.out.println(num);
num++;
} while (num <= 0);
}
}
```

Snippet 4:

```
public class OffByOneErrorForLoop {
public static void main(String[] args) {
for (int i = 1; i <= 10; i++) {
System.out.println(i);
}
// Expected: 10 iterations with numbers 1 to 10
// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9
}
}
```

// Error to investigate: What is the issue with the loop boundaries? How should the loop be adjusted to meet the expected output?

Here, the loop condition leads to iterate it 10 times but changing the loop condition i<10 we can get the expected output 1 to 9.

Correct code:

```
public class OffByOneErrorForLoop {
public static void main(String[] args) {
for (int i = 1; i < 10; i++) {
System.out.println(i);
}
// Expected: 10 iterations with numbers 1 to 10
// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9
}
}
```

Snippet 5:

```
public class WrongInitializationForLoop {
public static void main(String[] args) {
for (int i = 10; i >= 0; i++) {
System.out.println(i);
}
 }
 }
```

// Error to investigate: Why does this loop not print numbers in the expected order? What is the problem with the initialization and update statements in the `for` loop?

Here, the loop goes in infinite loop after execution because the updation i++ leads to it so, we need to change update statement to i--.

Correct Code:

```
public class WrongInitializationForLoop {
public static void main(String[] args) {
for (int i = 10; i >= 0; i--) {
System.out.println(i);
}
 }
 }
```

Snippet 6:

```
public class MisplacedForLoopBody {
public static void main(String[] args) {
for (int i = 0; i < 5; i++)
System.out.println(i);
System.out.println("Done");
 }
}
```

// Error to investigate: Why does "Done" print only once, outside the loop? How should the loop body be enclosed to include all statements within the loop?

As in this for loop doesn't have '{ }' so due to this only first statement is iterate in for loop.So, we need to add '{ }' to for loop to include all the statement with the loop.

Correct Code:

```
public class MisplacedForLoopBody {
public static void main(String[] args) {
for (int i = 0; i < 5; i++) {
System.out.println(i);
System.out.println("Done");
}
}
}
```

Snippet 7:

```
public class UninitializedWhileLoop {
public static void main(String[] args) {
int count;
while (count < 10) {
System.out.println(count);
count++;
}
```

```
}
}
```
// Error to investigate: Why does this code produce a compilation error? What needs to be done to initialize the loop variable properly?

Error: variable count might not have been initialized
while (count < 10) {
Here, the count variable is initialized & not declared so, we need to declare the count variable before using in while loop.

Correct Code:

```
public class UninitializedWhileLoop {
public static void main(String[] args) {
int count = 5;
while (count < 10) {
System.out.println(count);
count++;
}
}
}
```

Snippet 8:

```
public class OffByOneDoWhileLoop {
public static void main(String[] args) {
int num = 1;
do { System.out.println(num);
num--;
} while (num > 0);
}
}
```
// Error to investigate: Why does this loop print unexpected numbers? What adjustments are needed to print the numbers from 1 to 5?

Here, the condition num>0 & num-- leads to print unexpected number to print the number from 1 to 5 we need to do n++ & num<=5 in this.

Correct Code:

```
public class OffByOneDoWhileLoop {
public static void main(String[] args) {
int num = 1;
do { System.out.println(num);
num++;
} while (num <= 5);
```

```
}
}
```

Snippet 9:

```
public class InfiniteForLoopUpdate {
public static void main(String[] args) {
for (int i = 0; i < 5; i += 2) {
System.out.println(i);
}
}
}
```

// Error to investigate: Why does the loop print unexpected results or run infinitely? How should the loop update expression be corrected?

The update expression need to correct as i++. This loop prints unexpected results as:

i	i<5	s.o.p	i+=2	intially, i = 0;
0	yes	0	2	
2	yes	2	4	
4	yes	4	6	
6	no	loop terminate		

Correct Code:

```
public class InfiniteForLoopUpdate {
public static void main(String[] args) {
for (int i = 0; i < 5; i++) {
System.out.println(i);
}
}
}
```

Snippet 10:

```
public class IncorrectWhileLoopControl {
public static void main(String[] args) {
int num = 10;
while (num = 10) {
System.out.println(num);
num--;
}
}
}
```

// Error to investigate: Why does the loop execute indefinitely? What is wrong with the loop condition?

Error: incompatible types: int cannot be converted to boolean
while (num = 10) {
Here, in java, condition has the assignment operator = instead of the equality operator == to check the condition.

Correct Code:

```
public class IncorrectWhileLoopControl {
public static void main(String[] args) {
int num = 10;
while (num == 10) {
System.out.println(num);
num--;
}
}
}
```

Snippet 11:

```
public class IncorrectLoopUpdate {
public static void main(String[] args) {
int i = 0;
while (i < 5) {
System.out.println(i);
i += 2;
// Error: This may cause unexpected results in output
}
}
}
```

// Error to investigate: What will be the output of this loop? How should the loop variable be updated to achieve the desired result?

The update expression need to correct as i++. This loop prints unexpected results as:

i	i<5	s.o.p	i+=2	intially, i = 0;
0	yes	0	2	
2	yes	2	4	
4	yes	4	6	
6	no	loop terminate		

The output required is :

```
C:\Users\dell\Desktop\cdac\Day 3\Lab>java IncorrectLoopUpdate
0
1
2
3
4
```

Correct Code:

```
public class IncorrectLoopUpdate {
public static void main(String[] args) {
int i = 0;
while (i < 5) {
System.out.println(i);
i++;
}
}
}
```

Snippet 12:

```
public class LoopVariableScope {
public static void main(String[] args) {
for (int i = 0; i < 5; i++) {
int x = i * 2;
}
 System.out.println(x);
// Error: 'x' is not accessible here
}
}
```

// Error to investigate: Why does the variable 'x' cause a compilation error? How does scope

Error:

```
error: cannot find symbol
 System.out.println(x);
                    ^
  symbol:   variable x
  location: class LoopVariableScope
```

the variable x has the scope within the for loop only we can't access outside for loop to access we need to declare variable x outside for loop.

Correct Code:

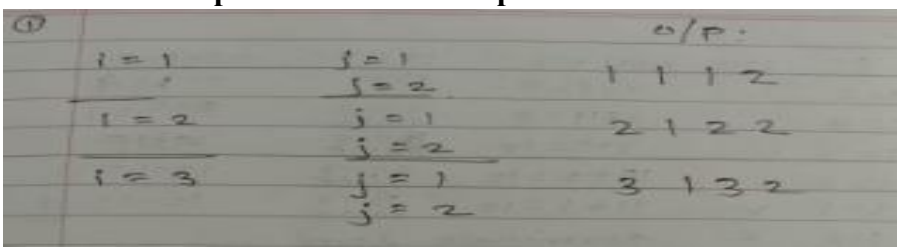
```
public class LoopVariableScope {
    public static void main(String[] args) {
        int x = 0; // Declare x outside the loop
        for (int i = 0; i < 5; i++) {
            x = i * 2; // Update x inside the loop
        }
        System.out.println(x); // Now x is accessible here
    }
}
```

SECTION 2: Guess the Output

Snippet 1:

```
public class NestedLoopOutput {
public static void main(String[] args) {
for (int i = 1; i <= 3; i++) {
for (int j = 1; j <= 2; j++) {
System.out.print(i + " " + j + " ");
}
System.out.println();
}
}
}
```

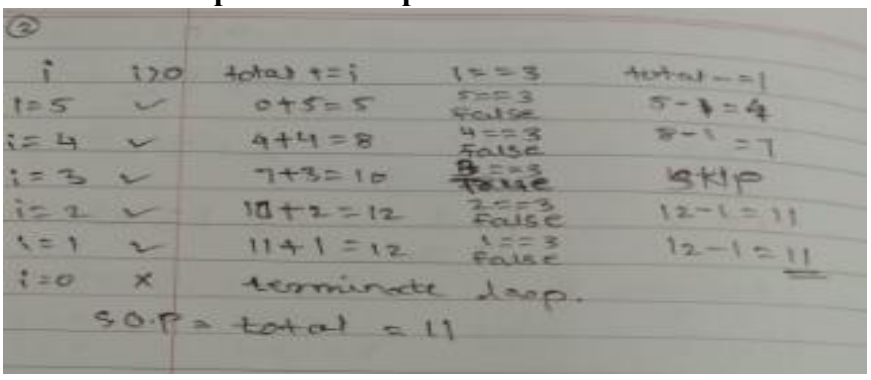
// Guess the output of this nested loop.



Snippet 2:

```
public class DecrementingLoop {
public static void main(String[] args) {
int total = 0;
for (int i = 5; i > 0; i--) {
total += i;
if (i == 3) continue;
total -= 1;
}
System.out.println(total);
}
}
```

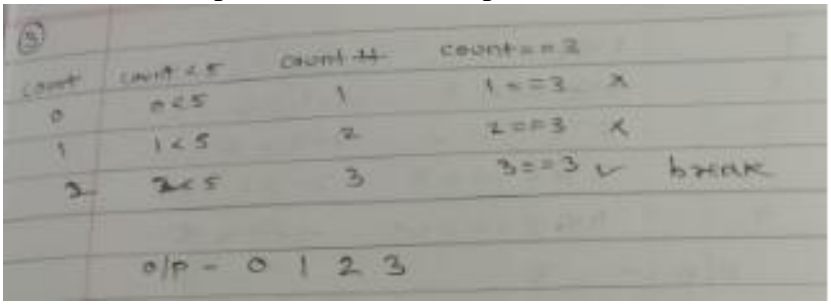
// Guess the output of this loop.



Snippet 3:

```
public class WhileLoopBreak {
public static void main(String[] args) {
int count = 0;
while (count < 5) {
System.out.print(count + " ");
count++;
if (count == 3)
break;
}
System.out.println(count);
}
}
```

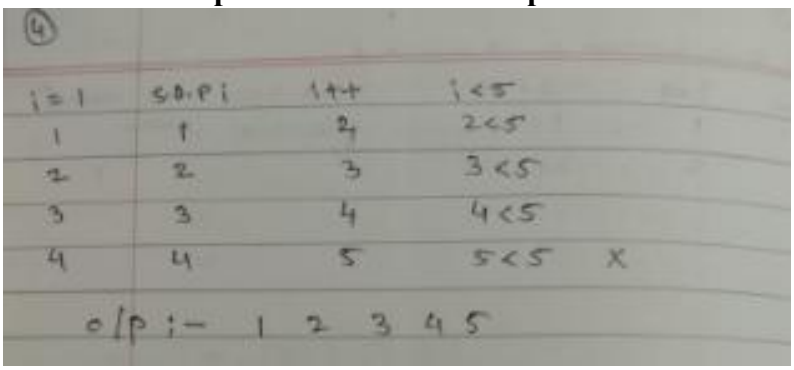
// Guess the output of this while loop.



Snippet 4:

```
public class DoWhileLoop {
public static void main(String[] args) {
int i = 1;
do {
System.out.print(i + " ");
i++;
} while (i < 5);
System.out.println(i);
}
}
```

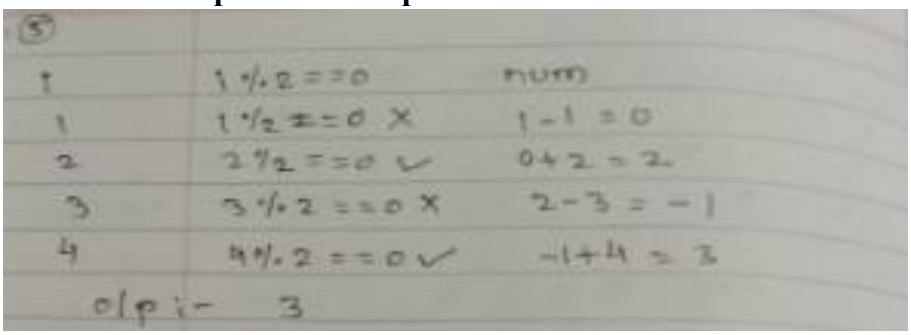
// Guess the output of this do-while loop.



Snippet 5:

```
public class ConditionalLoopOutput {
public static void main(String[] args) {
int num = 1;
for (int i = 1; i <= 4; i++) {
if (i % 2 == 0) {
num += i;
} else { num -= i;
}
}
System.out.println(num);
}
}
```

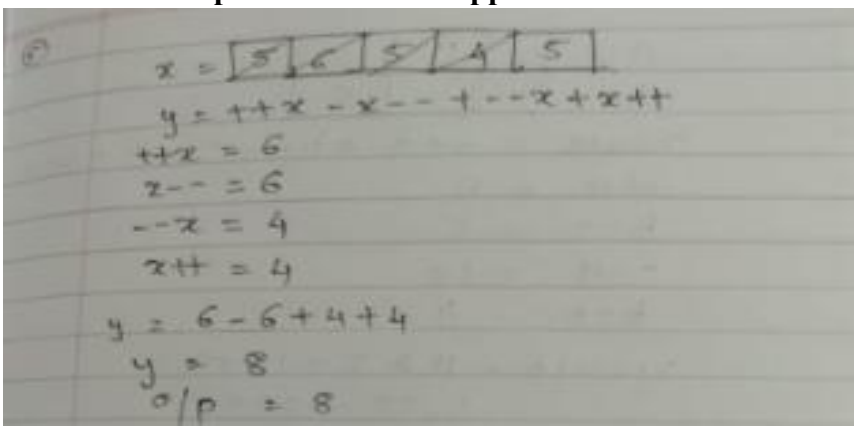
// Guess the output of this loop.



Snippet 6:

```
public class IncrementDecrement {
public static void main(String[] args) {
int x = 5;
int y = ++x - x-- + --x + x++;
System.out.println(y);
}
}
```

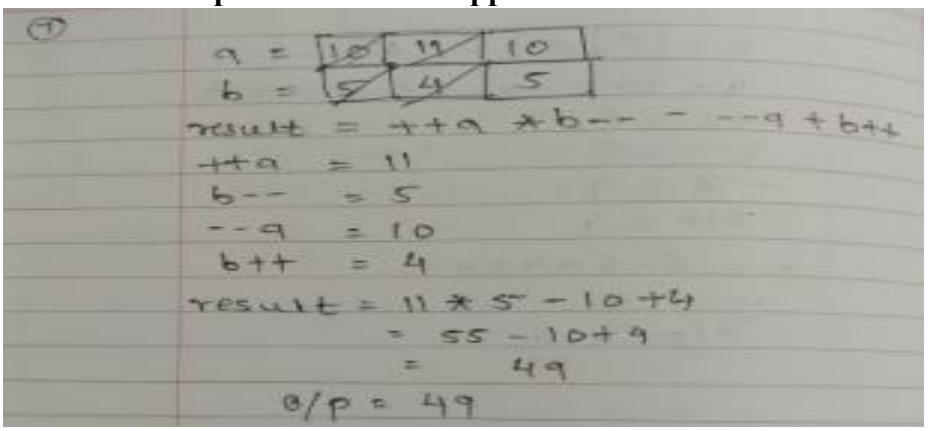
// Guess the output of this code snippet.



Snippet 7:

```
public class NestedIncrement {
public static void main(String[] args) {
int a = 10;
int b = 5;
int result = ++a * b-- - --a + b++; System.out.println(result);
}
}
```

// Guess the output of this code snippet.



Snippet 8:

```
public class LoopIncrement {
public static void main(String[] args) {
int count = 0;
for (int i = 0; i < 4; i++) {
count += i++ - ++i;
}
System.out.println(count);
}
}
```

// Guess the output of this code snippet.