Assignment 3

SECTION 1:

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?

- The Code will run infinite time until the system crash.

Why does this loop run infinitely?

- In loop I is initialise by 0.
- The Condition I<10 is always true.
- I is decrement like 0, -1 , -2 …
- So the for loop goes infinitely

How should the loop control variable be adjusted?

- The Change we can do is we can replace post decrement i-- to post increment i++.
- So instead of decrementing from Zero to infinity, Now we can increment i from 0 to 9 until the condition get false.

```
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?

```
IncorrectWhileCondition.java:4: error: incompatible types: int cannot be converted to boolean
while (count = 0) {
        ^
1 error
```

- The reason that tho loop not execute as expected because at while condition there is mistake that count is int.
- For while condition who accept only boolean.
- But due to wrong operator code is giving error.
- In above code for while condition (count = 0) Assignment operator is used.So count is assign 0 and it is int.
- To correct the code we have to use Relational operator which is '==' which will check count value is equal to 0.

Corrected code

```
public 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?

Output

```
0
1
2
3
4
.
.
Infinite
```

- No the loop is not executing once but it incrementing till infinity.
- To correct the code we have change the while condition like num < 10;

Corrected code

```
public class DoWhileIncorrectCondition {
	public static void main(String[] args) {
		int num = 0;
		do {
			System.out.println(num);
			num++;
		} while (num < 10);
	}
}
```

Output

```
0
1
2
3
4
.
.
9
```

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?

Actual Output

```
1
2
3
4
5
6
7
8
9
10
```

- The Actual Output is 1 - 10 but Expected output is 1 - 9.
- There we have to make a small change in above code at condition
- (int i = 1; i <= 10; i++) to (int I = 1; i<10 ; i++)
- By this change for loop will print 1 - 9.

Corrected Code

```
public class OffByOneErrorForLoop {
        public static void main(String[] args) {
                for (int i = 1; i <10; i++) { //  I have a change a condition
                        System.out.println(i);
                }
        }
}
```

Actual Output

```
1
2
3
4
5
6
7
8
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?

Actual Output
```
10
11
12
13
.
..
.
Infinity
```

- This loop not print in the expected order because there is some problem in increment
- We Have to use Decrement so it will give 10 - 0.

Corrected Code
```
public class WrongInitializationForLoop {
	public static void main(String[] args) {
		for (int i = 10; i >= 0; i--) {
			System.out.println(i);
		}
	}
}
```
Actual Output
```
10
9
8
7
6
5
4
3
2
1
0
```

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?

Actual ouput

```
0
1
2
3
4
Done
```

- The above code print 0 - 4 and done at last.· The absence of curly braces {} causes only the first statement (System.out.println(i);) to be part of the loop.
- · The second statement (System.out.println("Done");) is not inside the loop and executes only once after the loop finishes.

Corrected 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");
                }

        }
}
```

Actual ouput

```
0
Done
1
Done
2
Done
3
Done
4
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?

Compile TIme Error

```
UninitializedWhileLoop.java:4: error: variable count might not have been initialized
while (count < 10) {
    ^
1 error
```

- In above code count varriable only declare not initialized.
- So varriable count not get a memory and not also value.
- So it give Compile time error.

Corrected Error

```
public class UninitializedWhileLoop {
        public static void main(String[] args) {
                int count = 0; //initialized count with 0
                while (count < 10) {
                        System.out.println(count);
                        count++;
                }
        }
}
```

Actual ouput

```
0
1
2
3
4
5
6
7
8
9
```

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?

Actual ouput

```
1
```

- The output we get is due to we use decrement of num.
- Condition of while is also not right for expected output so we have to change to (num <=5)

Corrected code

```
public class OffByOneDoWhileLoop {
	public static void main(String[] args) {
		int num = 1;
		do {
			System.out.println(num);
			Num++;
		} while (num < 6);
	}
}
```

Actual ouput

```
0
1
2
3
4
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?

Actual ouput

```
0
2
4
```

- The loop Print unexpected result because in increment we are incrementing by 2
- To Fix this we have to use increment i++ so it will increment by 1.

Corrected code

```
public class InfiniteForLoopUpdate {
        public static void main(String[] args) {
                for (int i = 0; i < 5; i ++) { // Change to incremnt by 1
                        System.out.println(i);
                }
        }
}
```

Actual ouput

```
0
1
2
3
4
5
```

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?

```
IncorrectWhileLoopControl.java:4: error: incompatible types: int cannot be converted to
boolean
while (num = 10) {
       ^
1 error
```

- The reason that tho loop not execute as expected because at while condition there is mistake that count is int.
- For while condition who accept only boolean.
- But due to wrong operator code is giving error.
- In above code for while condition (count = 10) Assignment operator is used.So count is assign 0 and it is int.
- To correct the code we have to use Relational operator which is ‘==’ which will check count value is equal to 0.

Corrected 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?

Actual ouput

```
0
2
4
```

- The loop Print unexpected result because in increment we are incrementing by 2
- To Fix this we have to use increment i++ so it will increment by 1.

Corrected code

```
public class IncorrectLoopUpdate {
        public static void main(String[] args) {
                int i = 0;
                while (i < 5) {
                        System.out.println(i);
                        i += 1; // Error: This may cause unexpected results in output
                }
        }
}
```

Actual ouput

```
0
1
2
3
4
```

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

Compile Time Error

```
LoopVariableScope.java:6: error: cannot find symbol
System.out.println(x); // Error: 'x' is not accessible here
            ^
  symbol:   variable x
  location: class LoopVariableScope
1 error
```

- The variable x is declared inside the for loop block.
- Variables declared inside a block {} are only accessible within that block.
- Once the loop finishes, x goes out of scope and is no longer accessible.
- The statement System.out.println(x); is outside the loop, where x is undefined, causing a compilation error.

Corrected Code

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

Actual ouput

```
8
```

SECTION 2:

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

Dry Run

1. Iteration 1 (Outer Loop i = 1)
- Inner Loop j = 1 → Prints 1 1
- Inner Loop j = 2 → Prints 1 2

System.out.println(); moves to the next line.

2. Iteration 2 (Outer Loop i = 2)
- Inner Loop j = 1 → Prints 2 1
- Inner Loop j = 2 → Prints 2 2

System.out.println(); moves to the next line.

3. Iteration 3 (Outer Loop i = 3)
- Inner Loop j = 1 → Prints 3 1
- Inner Loop j = 2 → Prints 3 2

System.out.println(); moves to the next line.

Output

```
1 1 1 2
2 1 2 2
3 1 3 2
```

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.

Dry Run

```
total = 0
Iteration 1 (i = 5).   5>0 --> true
        total += 5--> total = 0+5 --> 5
        If 5 == 3 --> false
        total -=1 -->  total = 5-1 --> 4
        i--   i = 4
Iteration 1 (i = 4).   4>0 --> true
        total += 4--> total = 4+4--> 8
        If 4== 3 --> false
        total -=1 -->  total = 8-1 --> 7
        i--   i = 3
Iteration 1 (i = 3).  3>0 --> true
        total += total --> total = 7+3--> 10
        If 3== 3 --> true
        continue

        i--   i = 2
Iteration 1 (i = 2).  2>0 --> true
        total += total --> total = 10+2--> 12
        If 2== 0 --> false
        total -=1 -->  total = 12-1 --> 11
        i--   i = 1
Iteration 1 (i = 1).  1>0 --> true
        total += total --> total = 11+1--> 12
        If 1== 0 --> false
        total -=1 -->  total = 12-1 --> 11
        i--   i = 0
Iteration 1 (i = 0).  0>0 --> false

Tottal 11;
```

Output

```
11
```

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.

Dry Run

Initialization:

count = 0

Iteration 1 (count = 0)

    Print 0
    count++ → count = 1
    count != 3, so continue.

Iteration 2 (count = 1)

    Print 1
    count++ → count = 2
    count != 3, so continue.

Iteration 3 (count = 2)

    Print 2
    count++ → count = 3
    if (count == 3) break; → Loop terminates.

Print 3

Output

```
0 1 2 3
```

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.

Dry Run

Initialization:
i = 0
Iteration 1 (i= 0)
	Print 0
	i++ → i= 1
	1 < 5, true.
Iteration 2 (i = 1)
	Print 1
	i++ → i= 2
	2 < 5, true.
Iteration 3 (i = 2)
	Print 2
	i++ → i= 3
	3 < 5, true.
Iteration 3 (i = 3)
	Print 3
	i++ → i= 4
	4< 5, true.
Iteration 3 (i = 4)
	Print 4
	i++ → i= 5
	5< 5, false.
Out of loop

Print 5

Output

```
0 1 2 3 4 5
```

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

Dry Run

```
Initialization:
num  = 1
Iteration 1 (i= 1)    1 < =4, true.
        1 % 2==0    false
        num -= 1 →  0 ;
        i++ → i= 2
Iteration 2 (i= 2)    2 < =4, true.
        2 % 2==0    true
        num += 1 →  1;
        i++ → i= 3
Iteration 3 (i= 3)    3 < =4, true.
        3 % 2==0    false
        num -= 1 →  0 ;
        i++ → i= 4
Iteration 4 (i= 4)    4 < =4, true.
        4 % 2==0    true
        num += 1 →  1;
        i++ → i= 5
Iteration 5 (i = 5)   5< =4, false.
Out of loop
```

Print 1

Output

```
1
```

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.

Dry Run

Initialization:

x = 5 | 6 | 5 | 4 | 5

y = 6 -   6 + 4 + 4

Output

8

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.

Dry Run

Initialization:

a = 10 | 11 | 10 |

b = 5 | 4 | 5

Result = 11 * 5 - 10 + 4

Result = 49

Output

49

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.

Dry Run

Initialization:
count = 0
Iteration 1 (i= 0)   0 < 4, true.
        i = 0 | 1 | 2
        count +=0 - 2 → 0 -2 → -2
        i++
Iteration 2 (i= 3)   3 < 4, true.
        i = 3 | 4 | 5
        count +=3 - 5→ -2 -2 → -4
        i++
Iteration 3 (i= 6)   6 < 4, false.

Out of loop
Print -4

Output

```
-4
```