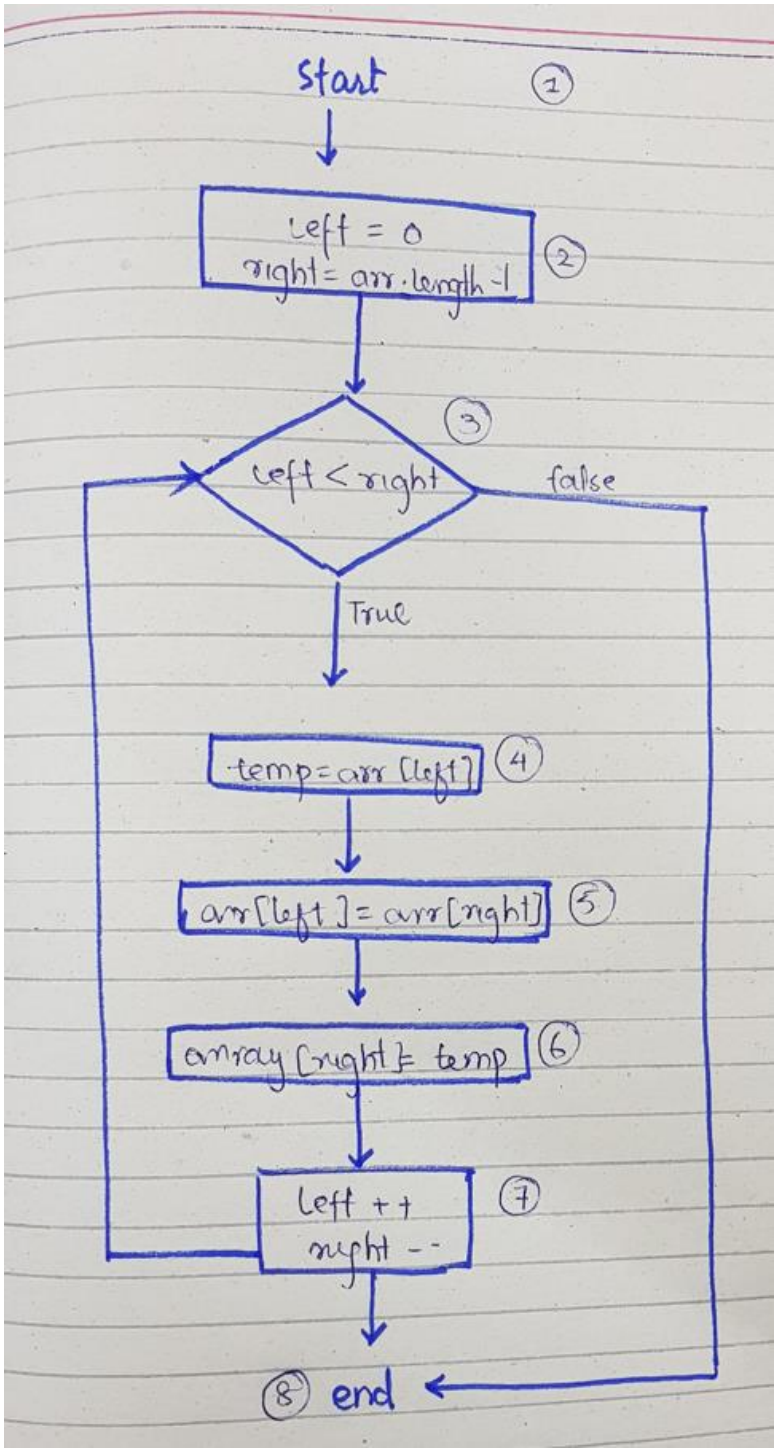
Contents

1. Algorithm: Reverse Array

Main.java × MainTest.java pom.xml (lab_Junit)

```java
package org.example;
//ReverseArray-Column(01)
public class Main {
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        reverseArray(array);
        for (int num : array) {
            System.out.print(num + " ");
        }
    }
    public static void reverseArray(int[] array) {  11 usages
        int left = 0, right = array.length - 1;
        while (left < right) {
            int temp = array[left];
            array[left] = array[right];
            array[right] = temp;
            left++;
            right--;
        }
    }
}
```

2. Control Flow Paths and Test Cases



3. Paths

1. Path 1: When the array has elements, the loop while (left < right) executes. **(Steps 1 → 2 → 3 → 4 → 5 → 6 → 7 → 2)**
2. Path 2: When the array is empty or has a single element, the loop while (left < right) does not execute. **(1-2-8)**

Step	Condition	Action	Next Step
1	Start	Initialize left to 0 and right to array.length - 1	2
2	left < right	True	3
3		Store array[left] in temp	4
4		Assign array[right] to array[left]	5
5		Assign temp to array[right]	6
6		Increment left by 1	7
7		Decrement right by 1	2
2	left < right	False	8
8	End	Exit method	-

4. Respective Test cases:

5.1 Test Cases for Path 1

Test Id	Description	Preconditions	Steps	Expected Result	Actual Result	Verdict
ReverseArray_TC1	Non-empty array (standard case)	None	Input: {1, 2, 3, 4, 5}	Output: {5, 4, 3, 2, 1}	{5, 4, 3, 2, 1}	Pass
ReverseArray_TC2	Array with negative values	None	Input: {-1, -2, -3, -4, -5}	Output: {-5, -4, -3, -2, -1}	{-5, -4, -3, -2, -1}	Pass
ReverseArray_TC3	Array with mixed positive/negative values	None	Input: {1, -2, 3, -4, 5}	Output: {5, -4, 3, -2, 1}	{5, -4, 3, -2, 1}	Pass
ReverseArray_TC4	Array with identical elements	None	Input: {1, 1, 1, 1}	Output: {1, 1, 1, 1}	{1, 1, 1, 1}	Pass
ReverseArray_TC5	Array with large numbers	None	Input: {100000, 200000, 300000, 400000, 500000}	Output: {500000, 400000, 300000, 200000, 100000}	{500000, 400000, 300000, 200000, 100000}	Pass

FA21-BSE-019
LABMIDTERM EXAM

Junit For Path 1

Main.java | MainTest.java | pom.xml (lab_Junit)

```java
package org.example;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class MainTest {

    // Path 1: Loop executes
    @Test
    void testNonEmptyArray() {
        int[] array = {1, 2, 3, 4, 5};
        Main.reverseArray(array);
        assertArrayEquals(new int[]{5, 4, 3, 2, 1}, array, message: "Array {1, 2, 3, 4, 5} should be reversed to {5, 4, 3, 2, 1}");
    }

    @Test
    void testArrayWithNegativeValues() {
        int[] array = {-1, -2, -3, -4, -5};
        Main.reverseArray(array);
        assertArrayEquals(new int[]{-5, -4, -3, -2, -1}, array, message: "Array {-1, -2, -3, -4, -5} should be reversed to {-5, -4, -3, -2, -1
    }

    @Test
    void testArrayWithMixedValues() {
        int[] array = {1, -2, 3, -4, 5};
        Main.reverseArray(array);
        assertArrayEquals(new int[]{5, -4, 3, -2, 1}, array, message: "Array {1, -2, 3, -4, 5} should be reversed to {5, -4, 3, -2, 1}");
    }

    @Test
    void testIdenticalElementsArray() {
        int[] array = {1, 1, 1, 1};
        Main.reverseArray(array);
        assertArrayEquals(new int[]{1, 1, 1, 1}, array, message: "Array {1, 1, 1, 1} should remain {1, 1, 1, 1} after reverse");
    }

    @Test
    void testArrayWithLargeNumbers() {
        int[] array = {100000, 200000, 300000, 400000, 500000};
        Main.reverseArray(array);
        assertArrayEquals(new int[]{500000, 400000, 300000, 200000, 100000}, array, message: "Array {100000, 200000, 300000, 400000, 500000} s
    }
```

5.2 Test Cases for Path 2

Test Id	Description	Preconditions	Steps	Expected Result	Actual Result	Verdict
ReverseArray_TC6	Empty array	None	Input: {}	Output: {}	{}	Pass
ReverseArray_TC7	Single-element array	None	Input: {10}	Output: {10}	{10}	Pass
ReverseArray_TC8	Array with zeros	None	Input: {0, 0, 0, 0, 0}	Output: {0, 0, 0, 0, 0}	{0, 0, 0, 0, 0}	Pass

Junit For Path 2

```java
// Path 2: Loop does not execute
@Test
void testEmptyArray() {
    int[] array = {};
    Main.reverseArray(array);
    assertArrayEquals(new int[]{}, array, message: "Empty array should remain empty after reverse");
}

@Test
void testSingleElementArray() {
    int[] array = {10};
    Main.reverseArray(array);
    assertArrayEquals(new int[]{10}, array, message: "Array with single element {10} should remain {10} after reverse");
}

@Test
void testArrayWithZeroes() {
    int[] array = {0, 0, 0, 0, 0};
    Main.reverseArray(array);
    assertArrayEquals(new int[]{0, 0, 0, 0, 0}, array, message: "Array {0, 0, 0, 0, 0} should remain {0, 0, 0, 0, 0} after reverse");
}
```

5.3 Failing Test Cases for Demonstration

Test Id	Description	Preconditions	Steps	Expected Result	Actual Result	Verdict
ReverseArray_Fail1	Non-empty array (standard case)	None	Input: {1, 2, 3, 4, 5}	Output: {5, 4, 3, 2, 1}	{1, 2, 3, 4, 5}	Fail
ReverseArray_Fail2	Single-element array	None	Input: {10}	Output: {10}	{20}	Fail

Junit For Failed TC

```java
// Failing test cases for demonstration
@Test
void testNonEmptyArray_Fail() {
    int[] array = {1, 2, 3, 4, 5};
    Main.reverseArray(array);
    assertArrayEquals(new int[]{1, 2, 3, 4, 5}, array, message: "Failing case: Array {1, 2, 3, 4, 5} should be incorrectly reversed to {1,
}

@Test
void testSingleElementArray_Fail() {
    int[] array = {10};
    Main.reverseArray(array);
    assertArrayEquals(new int[]{20}, array, message: "Failing case: Array with single element {10} should incorrectly change to {20}");
}
}
```

5. Output: Test Cases in JUnit

Following is the output of the test case written above: