IT 314 – Software Engineering

Lab 7

Akshat Jindal | 202201299

Merge Sort Code

Program Inspection

Question 1. How many errors are there in the program? Mention the errors you have identified.

The only error in this code was the syntax error that were coming from the incorrect use of left and right variables.

Question 2. Which category of program inspection would you find more effective?

For this question, the category A Data Reference Error was the most suited as it tackled the error of the wrongly referenced left and right variables.

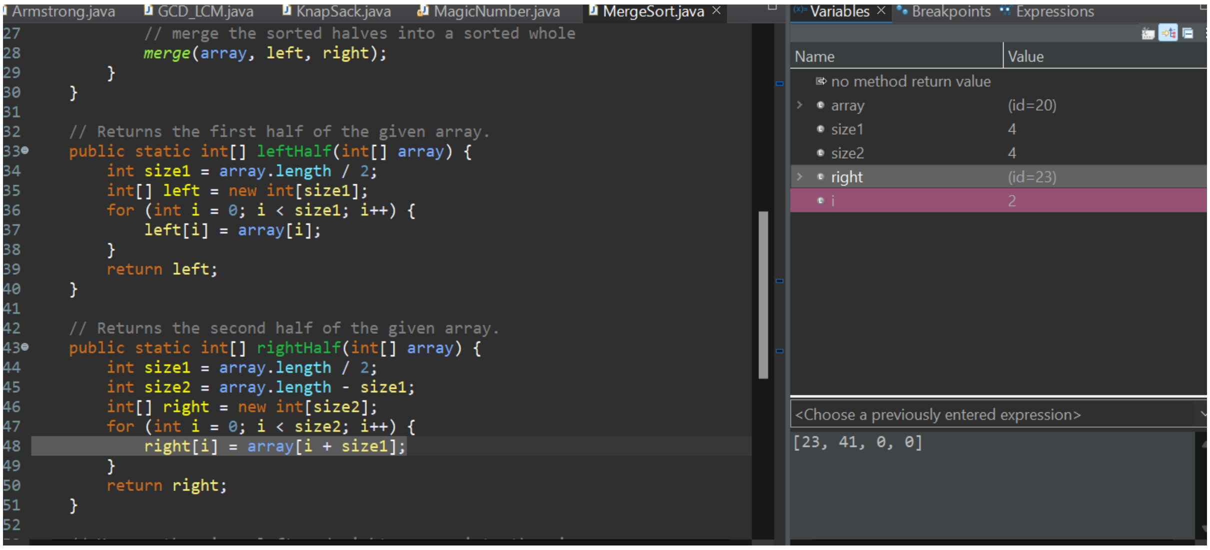
Question 3. Which type of error you are not able to identified using the program inspection?

In this example, all the errors in the document were identifiable using the program inspection method.

Question 4. Is the program inspection technique is worth applicable?

It was difficult in implementing the program inspection technique in this code. Due to the large length of the code, it was very difficult to check every item from the checklist by analysing the whole code.

Code Debugging



```java
            // merge the sorted halves into a sorted whole
            merge(array, left, right);
        }
    }

    // Returns the first half of the given array.
    public static int[] leftHalf(int[] array) {
        int size1 = array.length / 2;
        int[] left = new int[size1];
        for (int i = 0; i < size1; i++) {
            left[i] = array[i];
        }
        return left;
    }

    // Returns the second half of the given array.
    public static int[] rightHalf(int[] array) {
        int size1 = array.length / 2;
        int size2 = array.length - size1;
        int[] right = new int[size2];
        for (int i = 0; i < size2; i++) {
            right[i] = array[i + size1];
        }
        return right;
    }
```

Name	Value
no method return value	
array	(id=20)
size1	4
size2	4
right	(id=23)
i	2

<Choose a previously entered expression>

```
[23, 41, 0, 0]
```

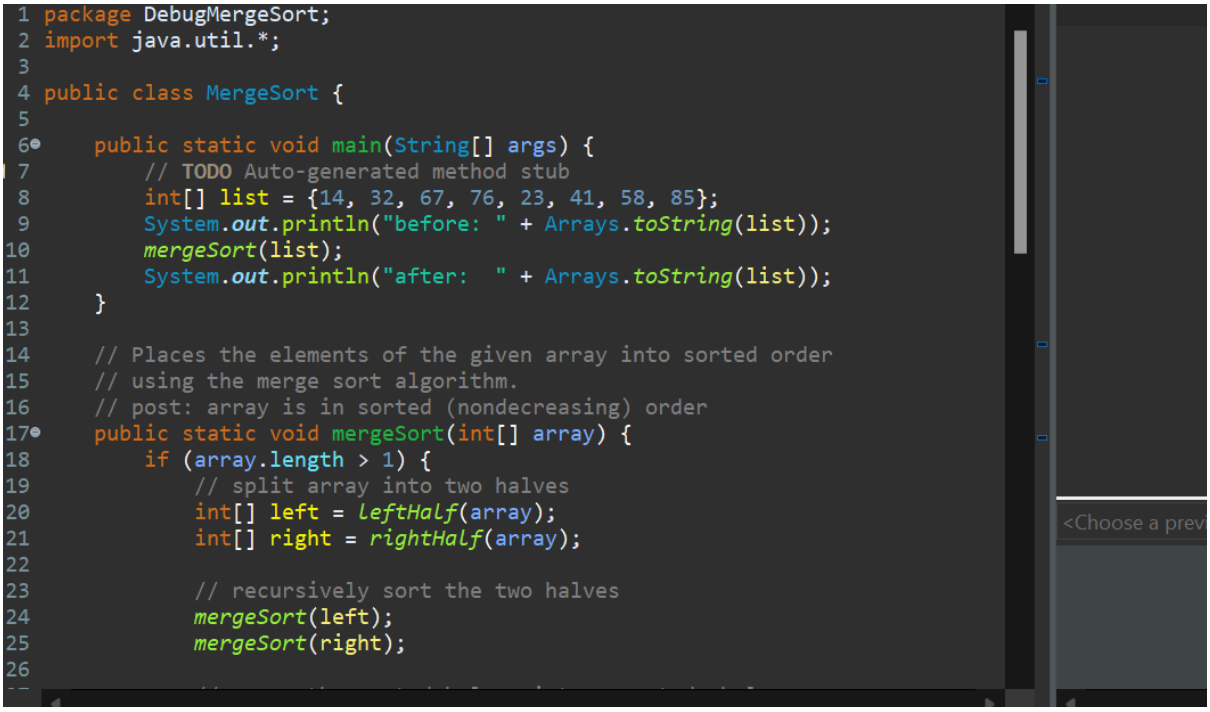
Question 1. How many errors are there in the program? Mention the errors you have identified.

There were many syntax errors in this code, which were corrected by passing the complete array and then calculating the left and right parts of the array.

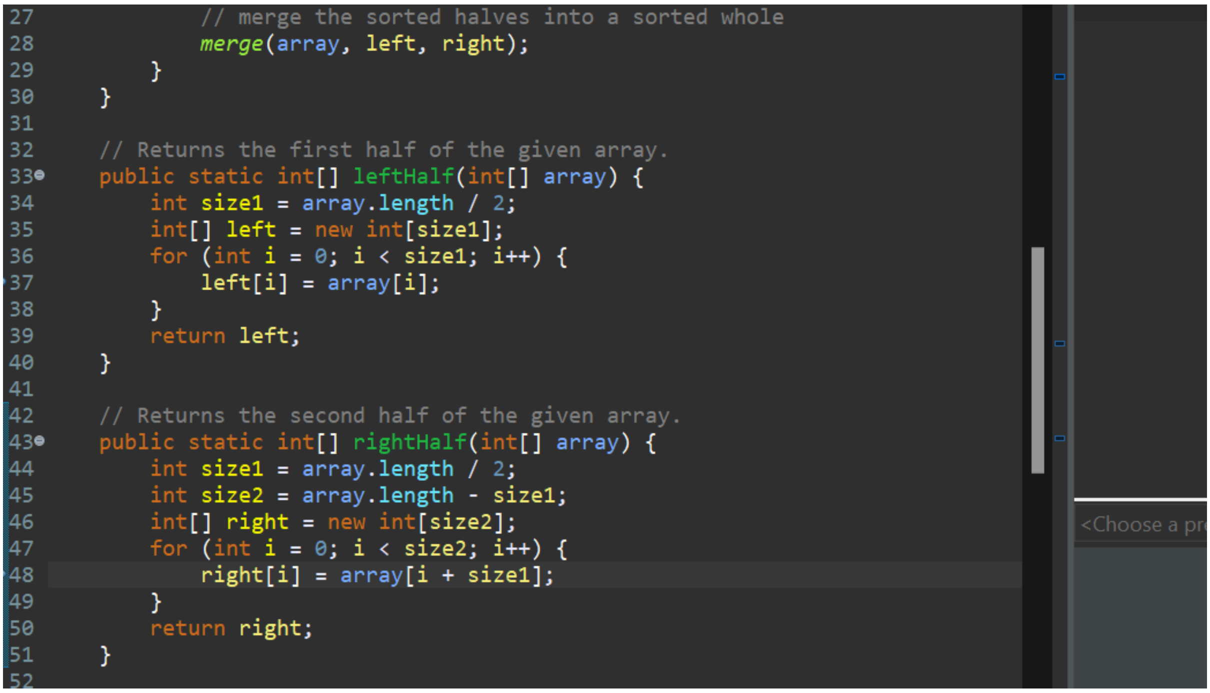
Question 2. How many breakpoints you need to fix those errors? What are the steps you have taken to fix the error you identified in the code fragment?

With 3 break points, inside the loop of calculating the left half, inside the loop of calculating the right half and inside the loop of calculating the merge array, we can find the mistake if there were any, regarding the execution, one iteration at a time.

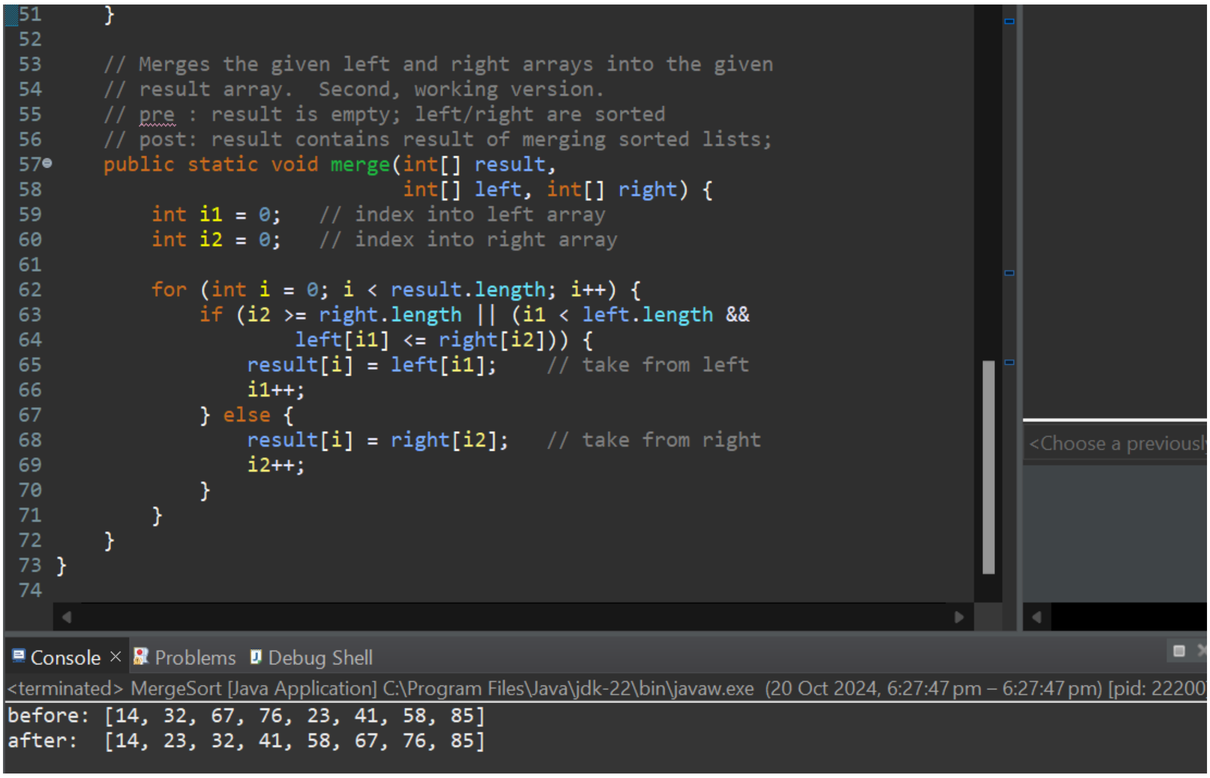
Question 3. Submit your complete executable code.



```
package DebugMergeSort;
import java.util.*;

public class MergeSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
        System.out.println("before: " + Arrays.toString(list));
        mergeSort(list);
        System.out.println("after:  " + Arrays.toString(list));
    }

    // Places the elements of the given array into sorted order
    // using the merge sort algorithm.
    // post: array is in sorted (nondecreasing) order
    public static void mergeSort(int[] array) {
        if (array.length > 1) {
            // split array into two halves
            int[] left = leftHalf(array);
            int[] right = rightHalf(array);

            // recursively sort the two halves
            mergeSort(left);
            mergeSort(right);

```



```
            // merge the sorted halves into a sorted whole
            merge(array, left, right);
        }
    }

    // Returns the first half of the given array.
    public static int[] leftHalf(int[] array) {
        int size1 = array.length / 2;
        int[] left = new int[size1];
        for (int i = 0; i < size1; i++) {
            left[i] = array[i];
        }
        return left;
    }

    // Returns the second half of the given array.
    public static int[] rightHalf(int[] array) {
        int size1 = array.length / 2;
        int size2 = array.length - size1;
        int[] right = new int[size2];
        for (int i = 0; i < size2; i++) {
            right[i] = array[i + size1];
        }
        return right;
    }

```



```java
    }

    // Merges the given left and right arrays into the given
    // result array.  Second, working version.
    // pre : result is empty; left/right are sorted
    // post: result contains result of merging sorted lists;
    public static void merge(int[] result,
                             int[] left, int[] right) {
        int i1 = 0;   // index into left array
        int i2 = 0;   // index into right array

        for (int i = 0; i < result.length; i++) {
            if (i2 >= right.length || (i1 < left.length &&
                    left[i1] <= right[i2])) {
                result[i] = left[i1];    // take from left
                i1++;
            } else {
                result[i] = right[i2];   // take from right
                i2++;
            }
        }
    }
}
```

```
<terminated> MergeSort [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (20 Oct 2024, 6:27:47 pm – 6:27:47 pm) [pid: 22200
before: [14, 32, 67, 76, 23, 41, 58, 85]
after:  [14, 23, 32, 41, 58, 67, 76, 85]
```