IT314

Software Engineering

Lab 7

Program Inspection, Debugging and Static Analysis

Name: Maulik Kansara

StudentId: 202201442

Lab Group: Group 5

II. CODE DEBUGGING: Debugging is the process of localizing, analyzing, and removing suspected errors in the code (Java code given in the .zip file)

Armstrong Problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Logic Error in Digit Extraction:

- Line: remainder = num / 10;
- Issue: The division num / 10 will result in the wrong digit because integer division returns the quotient, not the last digit. Instead, the remainder operator (%) should be used to extract the last digit.

Logic Error in Removing the Last Digit:

- Line: num = num % 10;
- Issue: The modulo operation is being used, which leaves the last digit, but we want to remove it. Instead, we should divide num by 10.

Issue with Armstrong Number Logic for General Cases:

- Issue: The program is using Math.pow(remainder, 3) to cube the digit. This works only for 3-digit numbers, but Armstrong numbers vary based on the number of digits. For instance, a 4-digit Armstrong number requires raising the digits to the power of 4.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

Two breakpoints:

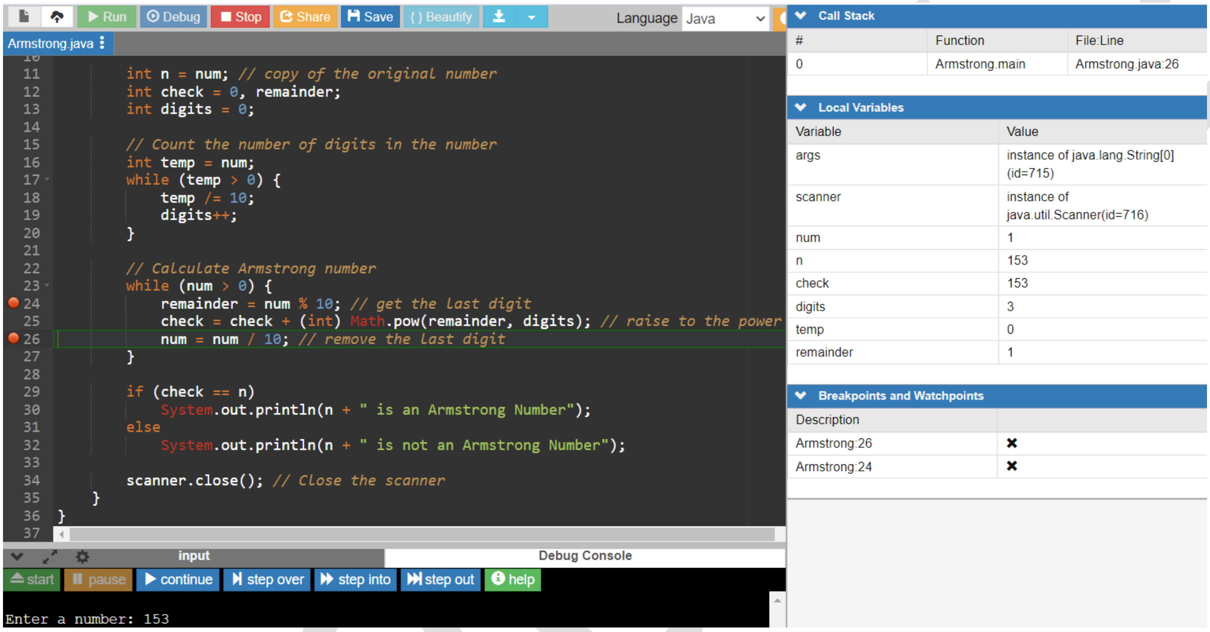
- Before calculating the remainder and updating check: To ensure the correct digit is being extracted.
- Before removing the last digit from num: To verify that the last digit is being correctly removed from the number

a. Steps to Fix the Errors:

1. Fix the logic for extracting the last digit:
   - Before: remainder = num / 10;
   - After: remainder = num % 10;

2. Fix the logic for removing the last digit:
    - Before: num = num % 10;
    - After: num = num / 10;
3. Fix the logic for the general case (multiple digits):
    - Find the number of digits in the input number n.
    - Use this count to raise each digit to the correct power when adding it to check.

Debugger window:



```java
        int n = num; // copy of the original number
        int check = 0, remainder;
        int digits = 0;

        // Count the number of digits in the number
        int temp = num;
        while (temp > 0) {
            temp /= 10;
            digits++;
        }

        // Calculate Armstrong number
        while (num > 0) {
            remainder = num % 10; // get the last digit
            check = check + (int) Math.pow(remainder, digits); // raise to the power
            num = num / 10; // remove the last digit
        }

        if (check == n)
            System.out.println(n + " is an Armstrong Number");
        else
            System.out.println(n + " is not an Armstrong Number");

        scanner.close(); // Close the scanner
    }
}
```

```
Enter a number: 153
```

Call Stack

#	Function	File:Line
0	Armstrong.main	Armstrong.java:26

Local Variables

Variable	Value
args	instance of java.lang.String[0] (id=715)
scanner	instance of java.util.Scanner(id=716)
num	1
n	153
check	153
digits	3
temp	0
remainder	1

Breakpoints and Watchpoints

Description	
Armstrong:26	✖
Armstrong:24	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

GCD and LCM problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Error in GCD Calculation Logic:

- Line: while(a % b == 0)
- Issue: The current condition a % b == 0 will terminate the loop if the numbers are divisible, which is incorrect. The loop should continue until b becomes 0 (i.e., a % b != 0).

Error in LCM Calculation Logic:

- Line: if(a % x != 0 && a % y != 0)
- Issue: The current condition is wrong because it checks whether a is not divisible by both x and y. Instead, we want a to be divisible by both x and y to find the LCM.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

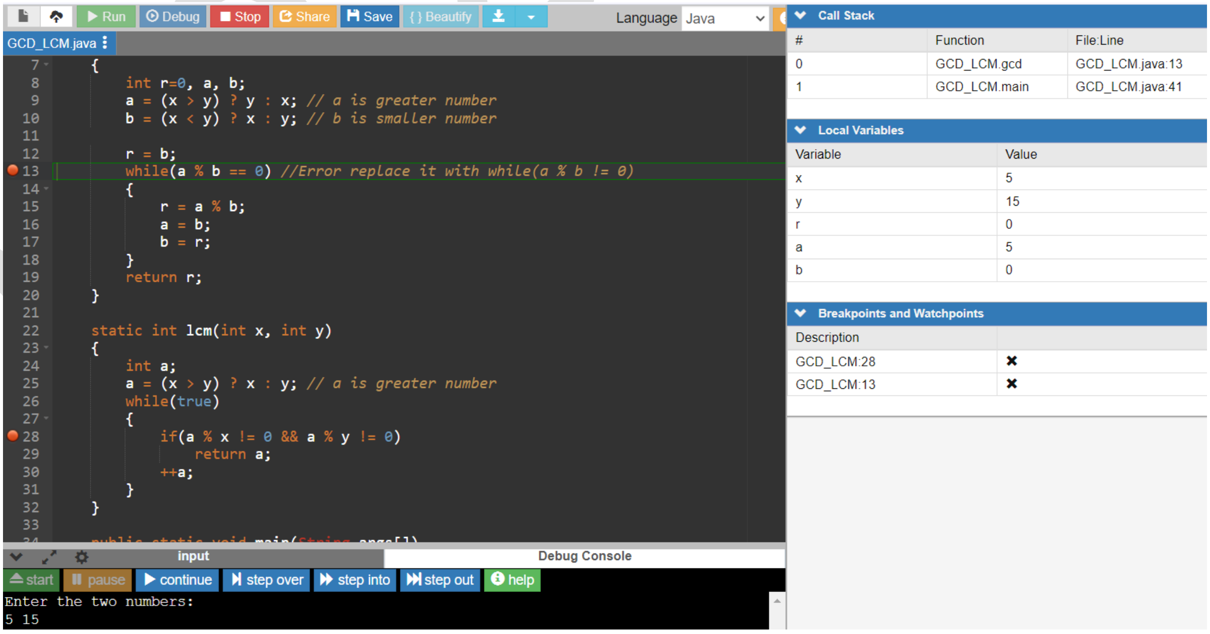
Two Breakpoints:

- Before the while loop in the GCD method: To ensure the loop runs until b becomes zero.
- Before checking the divisibility in the LCM method: To verify that the correct logic is used to find the LCM.

a. Steps to Fix the Errors:

1. Fix the GCD calculation logic:
   - Before: while (a % b == 0)
   - After: while (b != 0)
2. Fix the LCM calculation logic:
   - Before: if(a % x != 0 && a % y != 0)
   - After: if(a % x == 0 && a % y == 0)

Debugger Window:



```
 7      {
 8          int r=0, a, b;
 9          a = (x > y) ? y : x; // a is greater number
10          b = (x < y) ? x : y; // b is smaller number
11
12          r = b;
13          while(a % b == 0) //Error replace it with while(a % b != 0)
14          {
15              r = a % b;
16              a = b;
17              b = r;
18          }
19          return r;
20      }
21
22      static int lcm(int x, int y)
23      {
24          int a;
25          a = (x > y) ? x : y; // a is greater number
26          while(true)
27          {
28              if(a % x != 0 && a % y != 0)
29                  return a;
30              ++a;
31          }
32      }
33
34      public static void main(String args[])
```

```
Enter the two numbers:
5 15
```

Call Stack

#	Function	File:Line
0	GCD_LCM.gcd	GCD_LCM.java:13
1	GCD_LCM.main	GCD_LCM.java:41

Local Variables

Variable	Value
x	5
y	15
r	0
a	5
b	0

Breakpoints and Watchpoints

Description	
GCD_LCM:28	✖
GCD_LCM:13	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Knapsack Problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Indexing Error with n++:

- In the line int option1 = opt[n++][w];, using n++ increments n, which can lead to out-of-bounds errors and incorrect results. It should be opt[n-1][w]; instead to reference the correct previous state.

Incorrect Calculation for option2:

- In the line if (weight[n] > w) option2 = profit[n-2] + opt[n-1][w-weight[n]];, the condition should check if weight[n] <= w, and it should be option2 = profit[n] + opt[n-1][w-weight[n]]; (using profit[n] and adjusting for the correct index)

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

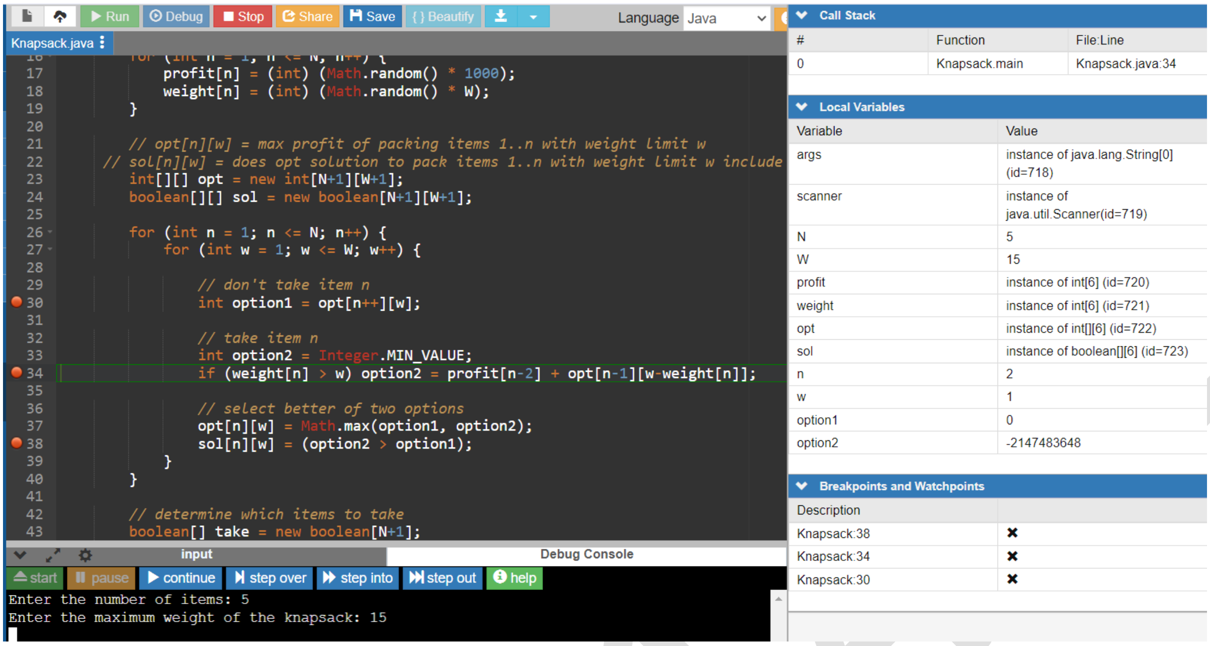
Three Breakpoints:

- After the line calculating option1 to ensure that n is correctly referenced as n-1.
- In the conditional block where option2 is calculated to check if it properly reflects the weight and profit of the current item.
- After the determination of items to take to validate that the solution correctly reflects the items chosen based on the sol matrix.

Steps to Fix the Errors:

1. Change n++ to n-1:
   - ○ Replace int option1 = opt[n++][w]; with int option1 = opt[n-1][w]; to prevent unintended incrementing.
2. Correct Calculation of option2:
   - ○ Modify the condition from if (weight[n] > w) to if (weight[n] <= w) and change option2 to profit[n] + opt[n-1][w-weight[n]];

Debugger Window:



```
16          for (int n = 1; n <= N; n++) {
17              profit[n] = (int) (Math.random() * 1000);
18              weight[n] = (int) (Math.random() * W);
19          }
20
21          // opt[n][w] = max profit of packing items 1..n with weight limit w
22      // sol[n][w] = does opt solution to pack items 1..n with weight limit w include
23          int[][] opt = new int[N+1][W+1];
24          boolean[][] sol = new boolean[N+1][W+1];
25
26          for (int n = 1; n <= N; n++) {
27              for (int w = 1; w <= W; w++) {
28
29                  // don't take item n
30                  int option1 = opt[n++][w];
31
32                  // take item n
33                  int option2 = Integer.MIN_VALUE;
34                  if (weight[n] > w) option2 = profit[n-2] + opt[n-1][w-weight[n]];
35
36                  // select better of two options
37                  opt[n][w] = Math.max(option1, option2);
38                  sol[n][w] = (option2 > option1);
39              }
40          }
41
42          // determine which items to take
43          boolean[] take = new boolean[N+1];
```

```
Enter the number of items: 5
Enter the maximum weight of the knapsack: 15
```

Call Stack

#	Function	File:Line
0	Knapsack.main	Knapsack.java:34

Local Variables

Variable	Value
args	instance of java.lang.String[0] (id=718)
scanner	instance of java.util.Scanner(id=719)
N	5
W	15
profit	instance of int[6] (id=720)
weight	instance of int[6] (id=721)
opt	instance of int[][6] (id=722)
sol	instance of boolean[][6] (id=723)
n	2
w	1
option1	0
option2	-2147483648

Breakpoints and Watchpoints

Description	
Knapsack:38	✖
Knapsack:34	✖
Knapsack:30	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Magic Number Check Problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Incorrect Loop Condition in the Inner Loop:

- The condition in the inner while loop should check if sum!=0, not sum == 0.

Wrong Operation for Summing Digits:

- The line s=s*(sum/10); should use addition instead of multiplication. It should be s = s + (sum % 10); to sum the digits properly.

Missing Semicolon:

- The line sum=sum%10 is missing a semicolon at the end.

Initial Value of sum:

- sum should be initialized to 0 at the start of the outer loop

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

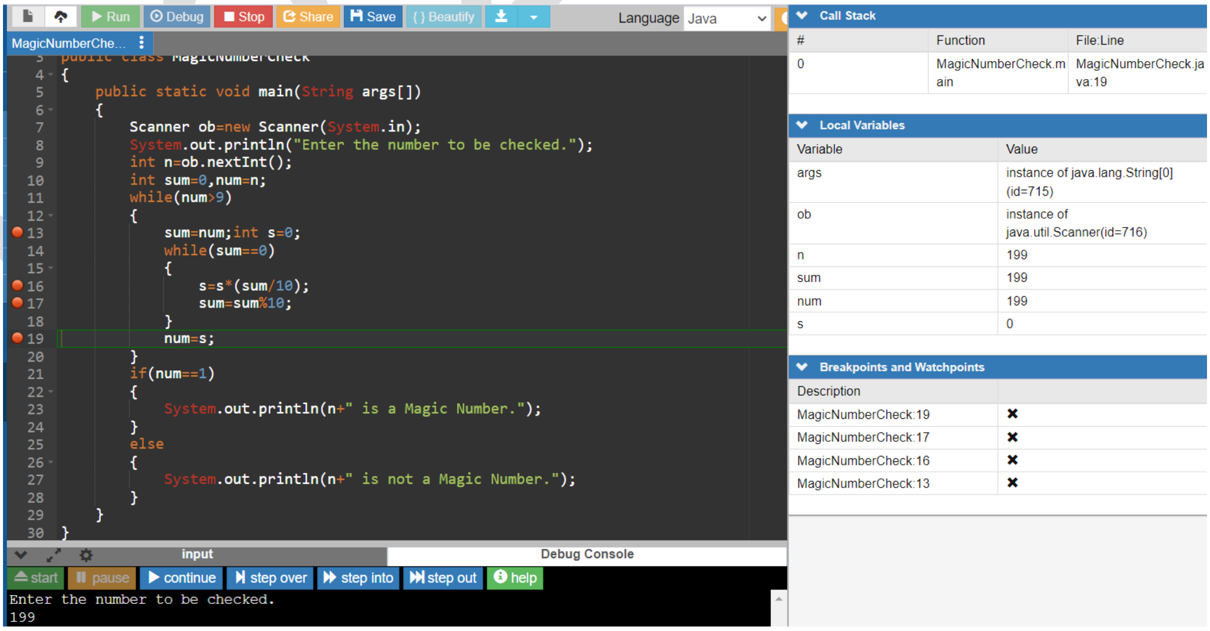
Four Breakpoints:

- After initializing sum to ensure it is reset for each outer loop iteration.
- Inside the inner loop to check the digit summing logic and ensure it uses addition.
- To check the loop condition in the inner while loop to ensure it processes digits correctly.
- To ensure proper termination of the loop and final output.

Steps to Fix the Errors:

1. Change the Inner Loop Condition:
   - Update while (sum == 0) to while (sum != 0).
2. Correct the Summation Logic:
   - Replace s = s * (sum / 10); with s = s + (sum % 10); to properly sum the digits.
3. Add Missing Semicolon:
   - Add a semicolon to the line sum = sum % 10;.
4. Initialize sum:
   - Ensure that sum is set to 0 at the start of the outer loop to avoid carrying over values from previous iterations.

Debugger Window:



```
public class MagicNumberCheck
{
    public static void main(String args[])
    {
        Scanner ob=new Scanner(System.in);
        System.out.println("Enter the number to be checked.");
        int n=ob.nextInt();
        int sum=0,num=n;
        while(num>9)
        {
            sum=num;int s=0;
            while(sum==0)
            {
                s=s*(sum/10);
                sum=sum%10;
            }
            num=s;
        }
        if(num==1)
        {
            System.out.println(n+" is a Magic Number.");
        }
        else
        {
            System.out.println(n+" is not a Magic Number.");
        }
    }
}
```

```
Enter the number to be checked.
199
```

Call Stack

#	Function	File:Line
0	MagicNumberCheck.main	MagicNumberCheck.java:19

Local Variables

Variable	Value
args	instance of java.lang.String[0] (id=715)
ob	instance of java.util.Scanner(id=716)
n	199
sum	199
num	199
s	0

Breakpoints and Watchpoints

Description	
MagicNumberCheck:19	✖
MagicNumberCheck:17	✖
MagicNumberCheck:16	✖
MagicNumberCheck:13	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Merge Sort Problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Incorrect Array Splitting:

- The leftHalf and rightHalf methods are called incorrectly. Instead of passing the array itself, they use array+1 and array-1, which are incorrect operations for array references.

Array Length Calculation:

- When calculating the left and right halves, the logic should involve slicing the original array without any offsets. The leftHalf and rightHalf methods should directly use the entire array for the calculations.

Postfix Increment/Decrement in merge:

- The parameters left++ and right-- in the merge method call should be left and right without any increments or decrements, as they should pass the entire arrays.

Boundary Condition in merge:

- The merge method should also handle the situation where the indices (i1, i2) might exceed the respective lengths of the left and right arrays properly. However, this is actually implemented correctly.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

Four Breakpoints:

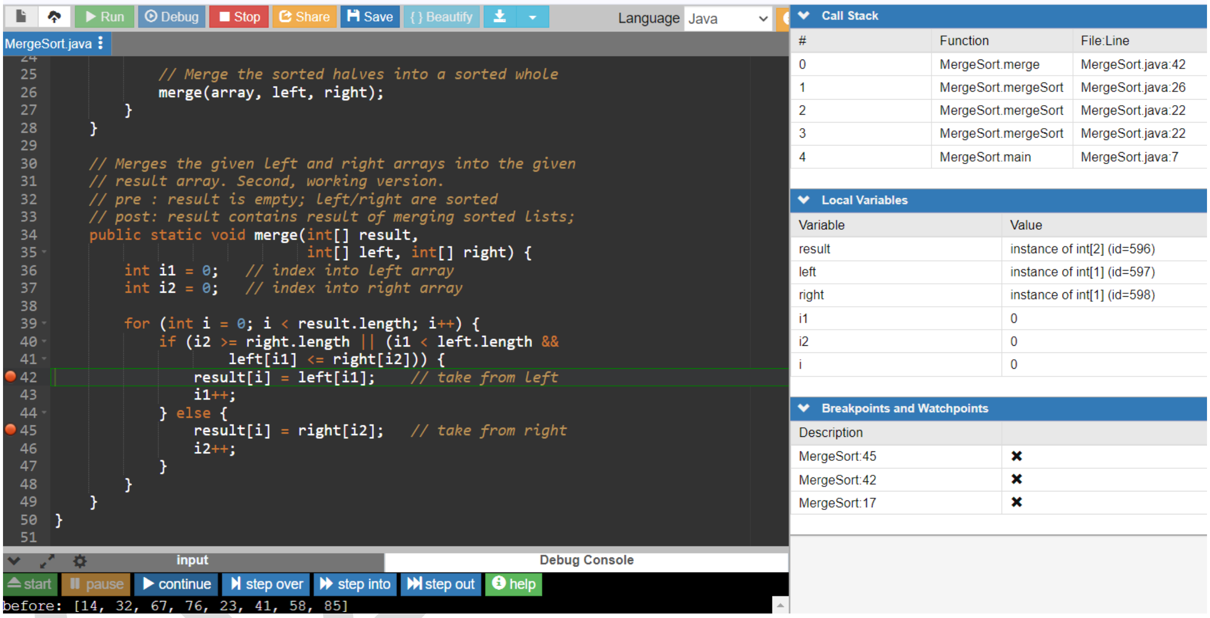
- At the point where the array is being split to ensure correct indexing.

- Inside the leftHalf method to verify that it correctly extracts the left half of the array.
- Inside the rightHalf method to ensure it extracts the right half of the array correctly.
- In the merge method to check if the merging logic is correctly implemented.

Steps to Fix the Errors:

1. Correct Array Splitting Logic:
   - Change the method calls to properly slice the original array without any offset.
2. Fix the Method Calls for Left and Right Halves:
   - Update the method calls in mergeSort to use proper slicing.
3. Remove Postfix Increment/Decrement in merge:
   - Ensure that the merge method is called with merge(array, left, right);.

Debugger Window:



```java
25             // Merge the sorted halves into a sorted whole
26             merge(array, left, right);
27         }
28     }
29 
30     // Merges the given left and right arrays into the given
31     // result array. Second, working version.
32     // pre : result is empty; left/right are sorted
33     // post: result contains result of merging sorted lists;
34     public static void merge(int[] result,
35                              int[] left, int[] right) {
36         int i1 = 0;   // index into left array
37         int i2 = 0;   // index into right array
38 
39         for (int i = 0; i < result.length; i++) {
40             if (i2 >= right.length || (i1 < left.length &&
41                     left[i1] <= right[i2])) {
42                 result[i] = left[i1];    // take from left
43                 i1++;
44             } else {
45                 result[i] = right[i2];   // take from right
46                 i2++;
47             }
48         }
49     }
50 }
51 
```

```
before: [14, 32, 67, 76, 23, 41, 58, 85]
```

Call Stack

#	Function	File:Line
0	MergeSort.merge	MergeSort.java:42
1	MergeSort.mergeSort	MergeSort.java:26
2	MergeSort.mergeSort	MergeSort.java:22
3	MergeSort.mergeSort	MergeSort.java:22
4	MergeSort.main	MergeSort.java:7

Local Variables

Variable	Value
result	instance of int[2] (id=596)
left	instance of int[1] (id=597)
right	instance of int[1] (id=598)
i1	0
i2	0
i	0

Breakpoints and Watchpoints

Description	
MergeSort:45	✖
MergeSort:42	✖
MergeSort:17	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Multiply Matrix Program:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Indexing Errors:

- In the multiplication loop, the indices used to access elements of the first and second matrices are incorrect. The indices are decremented improperly, which leads to an ArrayIndexOutOfBoundsException.

Sum Initialization:

- The sum variable should be reset to 0 at the beginning of the innermost loop, but it is currently reset at the end.

Matrix Multiplication Logic:

- The logic for calculating the matrix product is not following the correct formula. It should involve the element-wise multiplication of rows from the first matrix and columns from the second matrix.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

Three Breakpoints

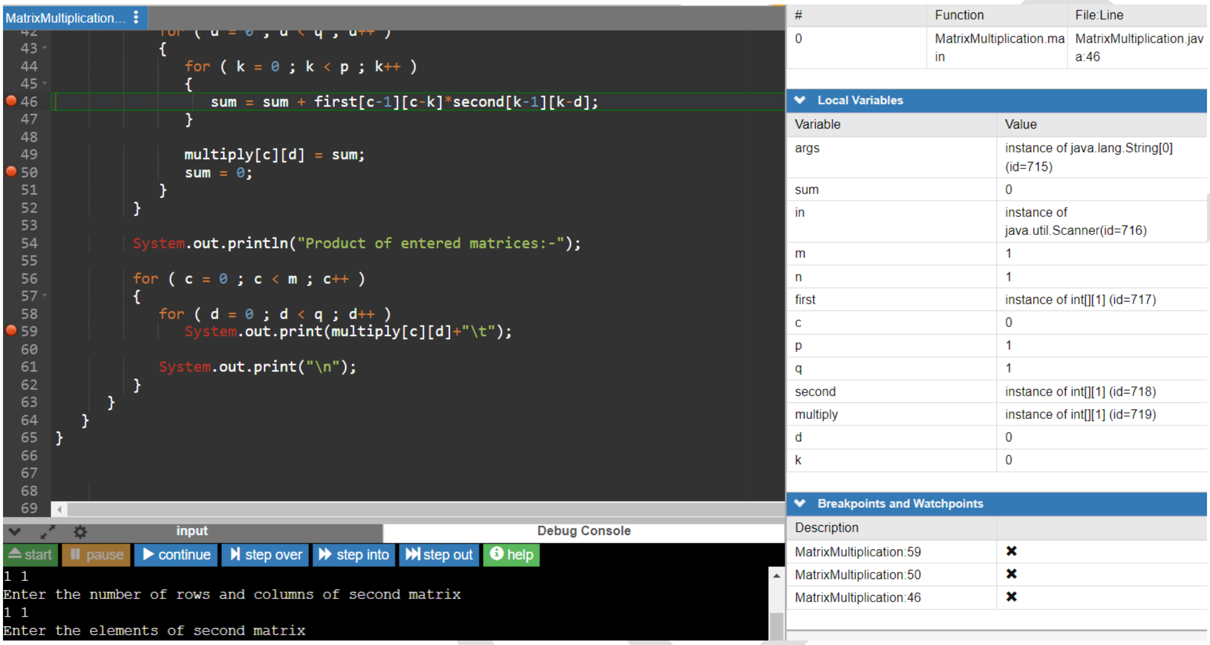
- Inside the loop for calculating the sum to check the indexing of first and second.
- At the start of the innermost loop to verify that sum is reset properly.
- After the multiplication loop to ensure that the resulting product matrix is printed correctly.

Steps to Fix the Errors:

1. Correct the Indexing:
   - Change the access of elements in the multiplication loop to avoid subtracting from c and k incorrectly.
2. Reset sum Correctly:
   - Move the sum = 0; initialization to the start of the innermost loop.
3. Fix the Multiplication Logic:

- Update the multiplication logic to correctly follow the formula for matrix multiplication: each element in the resulting matrix is the sum of the products of corresponding elements from the row of the first matrix and the column of the second matrix.

Debugger Window:



```
            for ( d = 0 ; d < q ; d++ )
            {
               for ( k = 0 ; k < p ; k++ )
               {
                  sum = sum + first[c-1][c-k]*second[k-1][k-d];
               }

               multiply[c][d] = sum;
               sum = 0;
            }
         }

         System.out.println("Product of entered matrices:-");

         for ( c = 0 ; c < m ; c++ )
         {
            for ( d = 0 ; d < q ; d++ )
               System.out.print(multiply[c][d]+"\t");

            System.out.print("\n");
         }
      }
   }
}
```

```
1 1
Enter the number of rows and columns of second matrix
1 1
Enter the elements of second matrix
```

#	Function	File:Line
0	MatrixMultiplication.main	MatrixMultiplication.java:46

Variable	Value
args	instance of java.lang.String[0] (id=715)
sum	0
in	instance of java.util.Scanner(id=716)
m	1
n	1
first	instance of int[][1] (id=717)
c	0
p	1
q	1
second	instance of int[][1] (id=718)
multiply	instance of int[][1] (id=719)
d	0
k	0

Description	
MatrixMultiplication:59	✖
MatrixMultiplication:50	✖
MatrixMultiplication:46	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Quardatic Probing:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Hash Function Logic:

- The hash function may return negative values, which can lead to an ArrayIndexOutOfBoundsException.

Insertion Logic:

- The insertion logic for quadratic probing incorrectly increments i using (i + h / h--). The proper quadratic probing should use i + h * h and increment h after using it.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

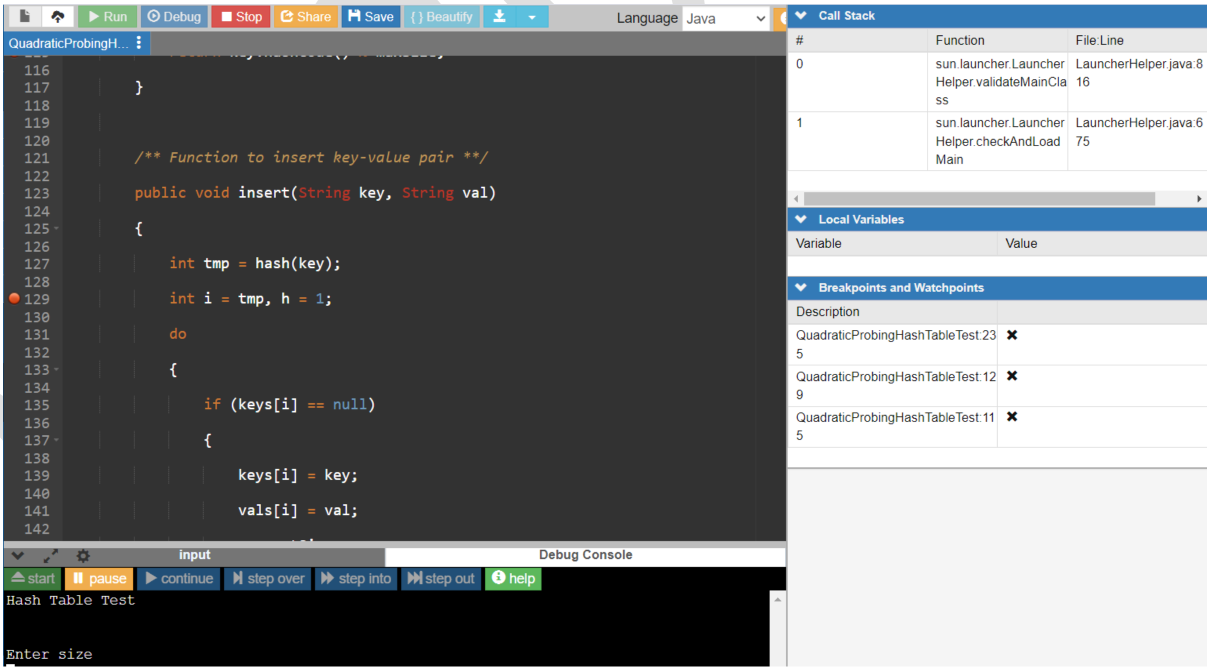
Three Breakpoints:

- Before the hash function to ensure the index returned is non-negative.
- Inside the insert method to check how i is being updated during probing.
- In the remove method to confirm that rehashing works correctly and maintains the correct size.

Steps to Fix the Errors:

1. Modify the Hash Function:
   - Ensure that the result of the hash function is always non-negative.
2. Correct the Insertion Logic:
   - Update the quadratic probing logic for the insertion and retrieval of keys.

Debugger Window:



```
    }

    /** Function to insert key-value pair **/
    public void insert(String key, String val)
    {
        int tmp = hash(key);
        int i = tmp, h = 1;
        do
        {
            if (keys[i] == null)
            {
                keys[i] = key;
                vals[i] = val;
```

```
Hash Table Test


Enter size
```

Call Stack

#	Function	File:Line
0	sun.launcher.LauncherHelper.validateMainClass	LauncherHelper.java:816
1	sun.launcher.LauncherHelper.checkAndLoadMain	LauncherHelper.java:675

Local Variables

Variable	Value

Breakpoints and Watchpoints

Description	
QuadraticProbingHashTableTest:235	✖
QuadraticProbingHashTableTest:129	✖
QuadraticProbingHashTableTest:115	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Sorting Array Problem:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Outer Loop Condition:

- The condition in the outer loop (for (int i = 0; i >= n; i++);) is incorrect. It should be i < n instead of i >= n, and the semicolon at the end of the loop should be removed.

Sorting Logic:

- The sorting logic currently tries to sort elements in descending order rather than ascending order due to the condition if (a[i] <= a[j]).

Printing Format:

- The printing of array elements has an issue where it prints a comma after the last element, which can be avoided.

Closing Scanner:

- The Scanner object should be closed at the end of its use to prevent resource leaks.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

Three BreakPoints:

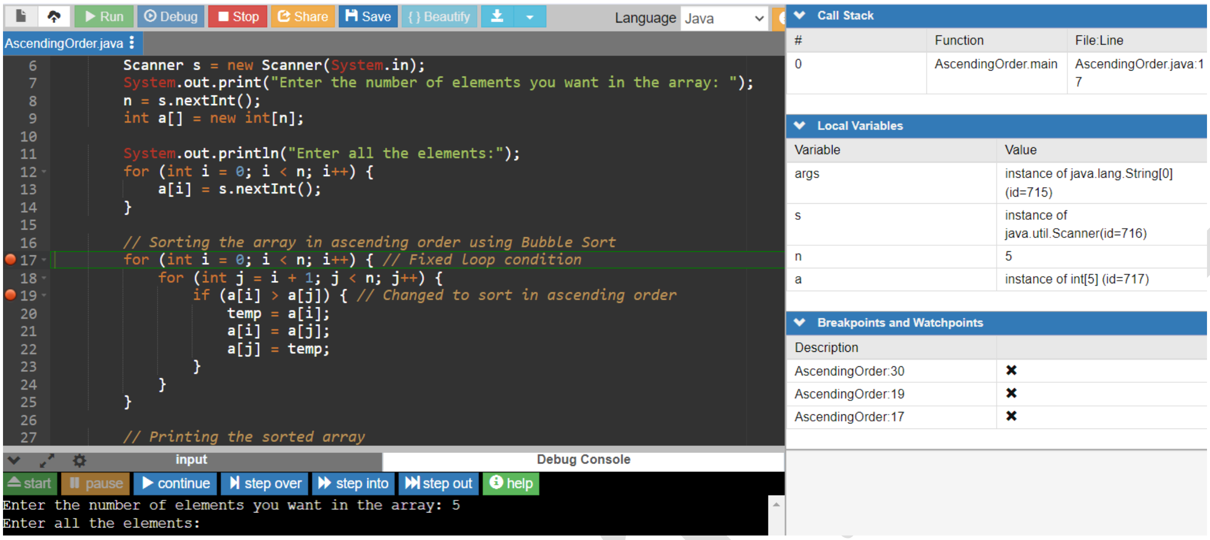
- Before the outer loop to check the initialization of i.
- Inside the sorting condition to verify the comparison logic.
- Just before printing the array to ensure proper formatting.

Steps to Fix the Errors:

1. Correct the Loop Conditions:
   - Update the outer loop condition to iterate properly through the array.
2. Adjust the Sorting Logic:
   - Change the comparison logic to ensure that elements are sorted in ascending order.
3. Improve the Output Formatting:

- Modify the print statement to avoid printing an extra comma.

Debugger Window:



```java
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the number of elements you want in the array: ");
        n = s.nextInt();
        int a[] = new int[n];

        System.out.println("Enter all the elements:");
        for (int i = 0; i < n; i++) {
            a[i] = s.nextInt();
        }

        // Sorting the array in ascending order using Bubble Sort
        for (int i = 0; i < n; i++) { // Fixed loop condition
            for (int j = i + 1; j < n; j++) {
                if (a[i] > a[j]) { // Changed to sort in ascending order
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }

        // Printing the sorted array
```

```
Enter the number of elements you want in the array: 5
Enter all the elements:
```

Call Stack

#	Function	File:Line
0	AscendingOrder.main	AscendingOrder.java:17

Local Variables

Variable	Value
args	instance of java.lang.String[0] (id=715)
s	instance of java.util.Scanner(id=716)
n	5
a	instance of int[5] (id=717)

Breakpoints and Watchpoints

Description	
AscendingOrder:30	✖
AscendingOrder:19	✖
AscendingOrder:17	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

Stack Implementation:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Push Method Logic:

- The logic in the push method decreases top before assigning the value to stack[top]. It should increase top instead.

Pop Method Logic:

- In the pop method, top is increased, but it should only be increased if you are popping a value from the stack (i.e., if top is not -1). When popping, the value at the top index should be ignored or returned.

Display Method Loop Condition:

- The loop condition in the display method should use < instead of > to iterate over the stack correctly.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

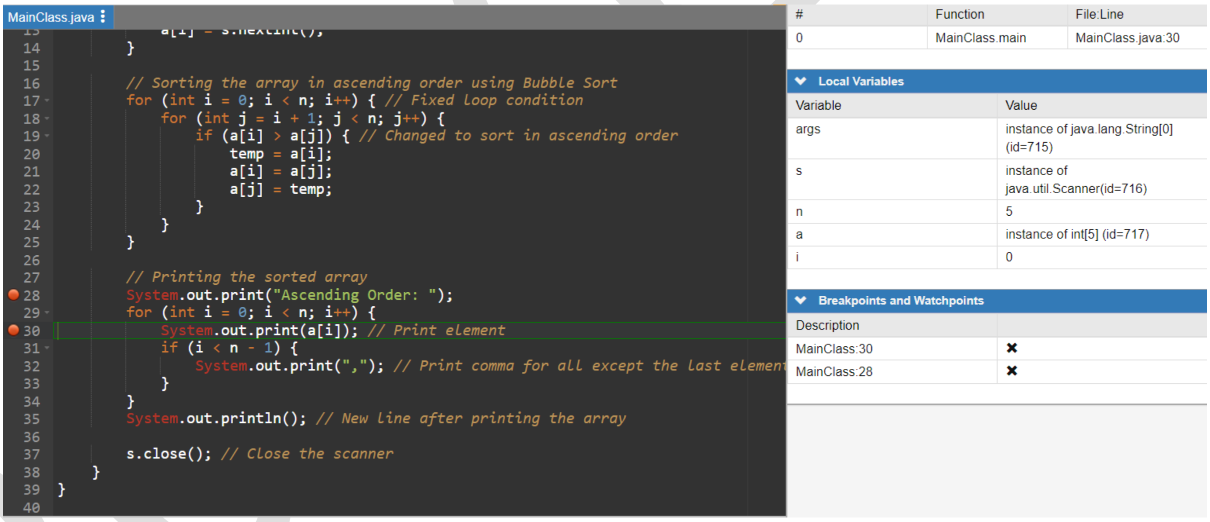
Three Breakpoints:

- In the push method to check the update of top.
- In the pop method to ensure the correct handling of top.
- In the display method to verify the loop condition.

Steps to Fix the Errors:

1. Correct the Logic in Push and Pop Methods:
   - Adjust the logic in both methods to correctly manage the top index.
2. Update the Display Method:
   - Fix the condition in the display loop to print all elements correctly.

Debugger Window:



```
            a[i] = s.nextInt();
        }

        // Sorting the array in ascending order using Bubble Sort
        for (int i = 0; i < n; i++) { // Fixed loop condition
            for (int j = i + 1; j < n; j++) {
                if (a[i] > a[j]) { // Changed to sort in ascending order
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }

        // Printing the sorted array
        System.out.print("Ascending Order: ");
        for (int i = 0; i < n; i++) {
            System.out.print(a[i]); // Print element
            if (i < n - 1) {
                System.out.print(","); // Print comma for all except the last element
            }
        }
        System.out.println(); // New line after printing the array

        s.close(); // Close the scanner
    }
}
```

#	Function	File:Line
0	MainClass.main	MainClass.java:30

Variable	Value
args	instance of java.lang.String[0] (id=715)
s	instance of java.util.Scanner(id=716)
n	5
a	instance of int[5] (id=717)
i	0

Description	
MainClass:30	✖
MainClass:28	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes files.

Tower of Hanoi:

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Incorrect Increment/Decrement in Recursive Calls:

- The expressions topN++ and inter-- in the recursive calls are incorrect. You should pass the same values without modifying them.

Incorrect Character Manipulation:

- Using from + 1 and to + 1 is incorrect when passing character arguments. Characters should not be incremented this way.

Missing Recursive Case for Moving the Remaining Disks:

- The logic in the doTowers method should handle the recursion correctly, and the way to move the disks back is not done properly.

Q2 How many breakpoints you need to fix those errors?

a. What are the steps you have taken to fix the error you identified in the code fragment?

Answer:

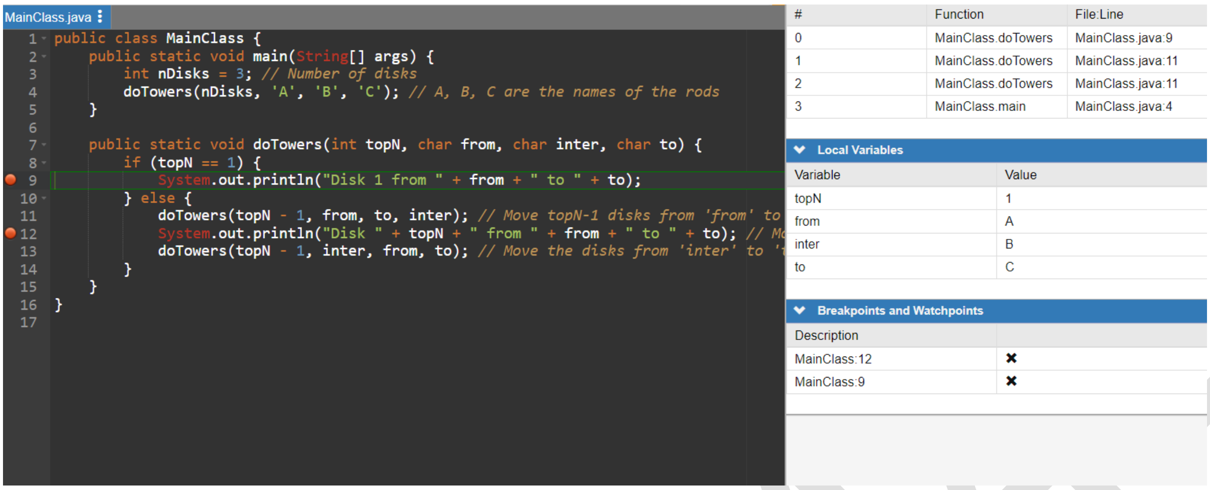
Three Breakpoints:

- In the doTowers method to ensure that topN is decremented correctly when making the recursive calls.
- In the method to check that characters (from, to, and inter) are passed without modifying them.
- To validate that the logic for moving the disks is properly executed.

Steps to Fix the Errors:

1. Correct the Recursive Call Parameters:
   - Use topN - 1 instead of topN++ for the first recursive call.
   - Use from, inter, and to directly without modifying them in the calls.
2. Simplify Character Handling:
   - Avoid attempting to increment or decrement character parameters. Use them as-is.

Debugger Window:



MainClass.java

```java
public class MainClass {
    public static void main(String[] args) {
        int nDisks = 3; // Number of disks
        doTowers(nDisks, 'A', 'B', 'C'); // A, B, C are the names of the rods
    }

    public static void doTowers(int topN, char from, char inter, char to) {
        if (topN == 1) {
            System.out.println("Disk 1 from " + from + " to " + to);
        } else {
            doTowers(topN - 1, from, to, inter); // Move topN-1 disks from 'from' to
            System.out.println("Disk " + topN + " from " + from + " to " + to); // Mo
            doTowers(topN - 1, inter, from, to); // Move the disks from 'inter' to '
        }
    }
}
```

#	Function	File:Line
0	MainClass.doTowers	MainClass.java:9
1	MainClass.doTowers	MainClass.java:11
2	MainClass.doTowers	MainClass.java:11
3	MainClass.main	MainClass.java:4

Local Variables

Variable	Value
topN	1
from	A
inter	B
to	C

Breakpoints and Watchpoints

Description	
MainClass:12	✖
MainClass:9	✖

Q3 Submit your complete executable code?

Answer: In corrected_java_codes file.

I. PROGRAM INSPECTION:

GitHub Repository Link (1601 LOC):
https://github.com/godotengine/godot/blob/master/editor/code_editor.cpp

Programming Language: CPP

Q1 How many errors are there in the program? Mention the errors you have identified.

Answer:

Category A: Data Reference Errors:

- Uninitialized Variables:
  In the constructor CodeTextEditor::CodeTextEditor(), some member variables like zoom_factor, code_complete_enabled, and idle are used without clear initialization.
- Array Subscript Bounds:
  The method goto_next_bookmark() and goto_prev_bookmark() check indices of PackedInt32Array bmarks without ensuring that the accessed index is within bounds.

Category B: Data-Declaration Errors

- Explicit Declaration of Variables:
  The snippet does not show explicit declarations for variables such as completion_font_color, code_complete_enabled, and find_replace_bar. If these variables are not declared elsewhere in the class, it could lead to confusion regarding their types and scopes.
- Proper Initialization:
  There is no indication that the variables being set (e.g., text_editor, code_complete_timer, idle) are properly initialized. Without context on their initialization, we cannot be certain if they are valid or not before being used.

Category C: Computation Errors:

- Inconsistent Data Types:
  In the function int GotoLineDialog::get_line() const, the method line->get_text().to_int() is used to convert a string to an integer. If the content of the text field is not a valid integer (e.g., it contains characters), this could lead to unexpected results or errors. Proper validation should be implemented.

Category D: Comparison Errors

- Mixed-mode comparisons or comparisons between variables of different types:
  There is a comparison in GotoLineDialog::ok_pressed: if (line_number < 0 || line_number >= text_editor->get_line_count()) line_number is an int, but it's unclear what get_line_count() returns (it might return an unsigned integer). If get_line_count() returns a type other than int, this comparison may lead to issues depending on the underlying conversion rules.

Category E: Control-Flow Errors

- No Error From this Category Found before there id no unaccounted flow altering statements.

Category F: Interface Errors

- Potential Type Mismatches with EDITOR_GET:
  The usage of EDITOR_GET to retrieve settings may return values of varying types (e.g., int, bool, String), and casting is applied to some returns (e.g., (int)EDITOR_GET(...)). If EDITOR_GET returns an unexpected type, it may lead to runtime errors.
- Undefined Behavior on Input Parameters:
  Functions like toggle_inline_comment assume the input parameters (e.g., delimiter) are valid without checking them.
  Ensure that the delimiter isn't an empty string before attempting to insert.
- Missing Parameter Checks:

Before using values from EDITOR_GET, there are no checks for whether these settings exist or are valid.

Category G: Input / Output Errors

- No Error From this Category Found because the code does not handle file.

Category H: Other Checks

- Potential Missing Error Handling:
  In functions like insert_final_newline, consider checking if final_line is negative before accessing it. This will prevent runtime errors if the text editor is empty.
- Unused Variables:
  The variable code_complete_enabled is retrieved using EDITOR_GET, but its usage is unclear. If it is not used elsewhere, it could be removed or its intended purpose clarified.
- Cross-reference Listing:
  The code has a variable line_label, which is initialized to nullptr in the constructor and does not seem to be used anywhere in the class. This could be flagged by a cross-reference check as it is never referenced.
- Robustness:
  The program does not seem to validate the input for the line variable before converting it to an integer in get_line(). Although there is validation in the ok_pressed() method, where it checks if line_number is within the valid range, this could potentially lead to a failure if invalid input is passed to get_line(). Additionally, functions like _search() do not seem to handle cases where the search text might be empty, which could cause unintended behavior

Q2 Which category of program inspection would you find more effective?

Answer:

According to me, Category A and C of Program inspection would be more effective, because:

Category A: Data Reference Errors:

- Data reference issues, such as uninitialized variables, out-of-bounds array access, and dangling references, can lead to critical run-time errors and undefined behavior. These types of errors are frequent, and inspecting them can prevent severe bugs early on, especially in languages with manual memory management like C/C++.

Category C: Computation Errors:

- Computation errors, such as overflow, mixed-mode arithmetic, and division by zero, are critical for ensuring the correctness of logic in programs. These issues are often harder to detect by compilers and can result in incorrect program outputs or even crashes in certain cases

Q3 Which type of error you are not able to identified using the program inspection?

Answer:

Error that can be difficult to identified using the program inspection:

- Run-Time Environment or External Dependency Errors: Errors related to the run-time environment, such as issues with memory allocation on specific hardware, unavailable resources, or race conditions in multithreaded programs, are hard to catch through inspection. These often require dynamic testing in the actual execution environment.
- Concurrency Errors (Race Conditions, Deadlocks): Concurrency issues like race conditions or deadlocks often depend on the specific timing of operations and may not be apparent in code inspection. They typically manifest only under certain execution conditions, making them hard to identify without tools or tests designed for concurrent execution.
- Performance Issues (Memory Leaks, Inefficiency): While program inspection can identify potential performance bottlenecks (e.g., inefficient algorithms), issues like memory leaks or inefficient memory usage are often detected more reliably through dynamic analysis (profiling or memory management tools).

Q4 Is the program inspection technique is worth applicable?

Answer:

Yes, program inspection techniques are worth applying for this type of project, for several reasons:

- Code inspection allows early identification of potential bugs before the software reaches the testing or deployment stages. Issues like missing input validation, null checks, and code duplication can be caught early.
- Regular inspections lead to better code organization, clearer logic, and improved maintainability. For instance, refactoring redundant code or adding necessary input validations.
- Code inspection gives us a deeper understanding of the system's internal workings, which can help us identify potential areas for optimization.

III. Static Analysis Tools

Choose a static analysis tool (in Java, Python, C, C++) in any programming language of your interest and identify the defects. You can also choose your own code fragment from GitHub (more than 2000 LOC) in any programming language to perform static analysis.

Answer:

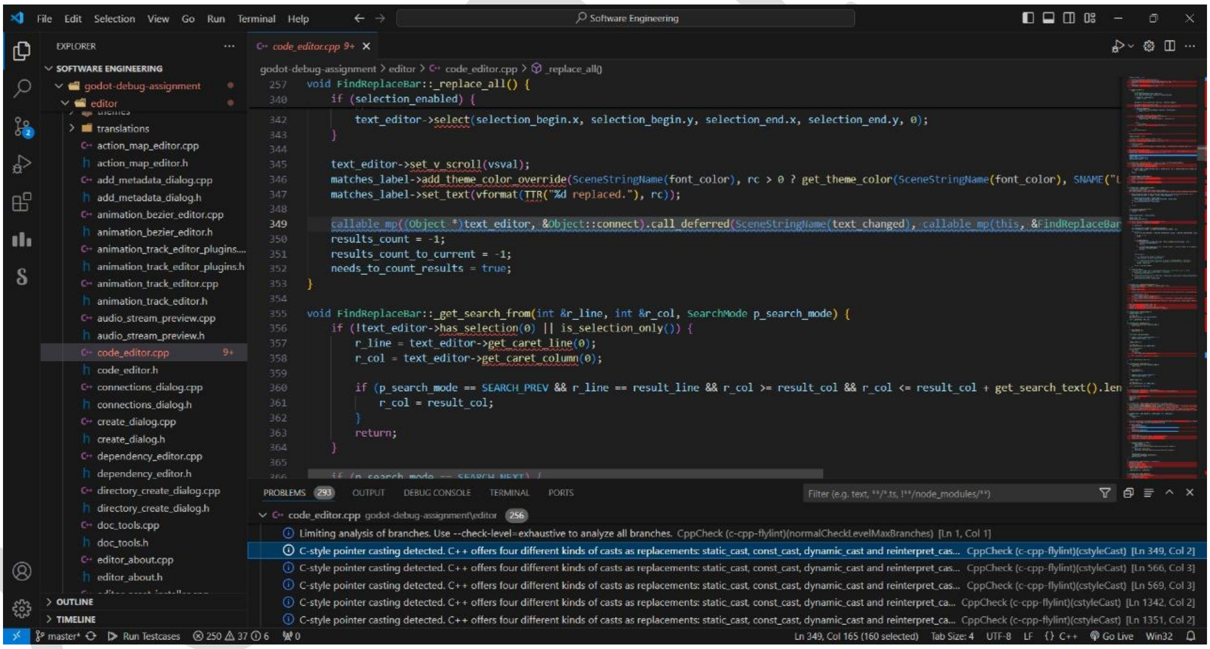
Programming Language: **CPP**

Tool Used: **CppCheck**

The static analysis tool is showing 6 informational messages. Additionally, Intellisense is highlighting errors caused by missing header file inclusions.

The Excel file of CppCheck is included in the folder.

Snippets of static analysis tool output:



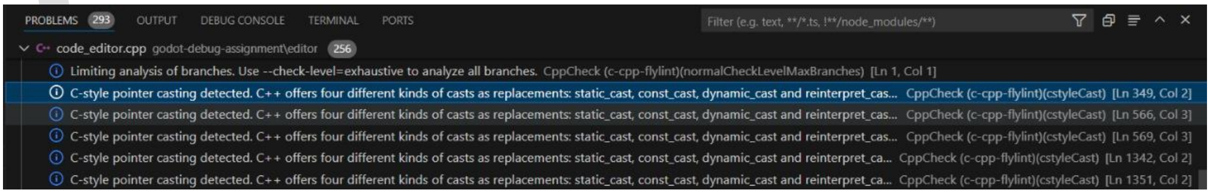
```cpp
void FindReplaceBar::_replace_all() {
    if (selection_enabled) {
        text_editor->select(selection_begin.x, selection_begin.y, selection_end.x, selection_end.y, 0);
    }

    text_editor->set_v_scroll(vsval);
    matches_label->add_theme_color_override(SceneStringName(font_color), rc > 0 ? get_theme_color(SceneStringName(font_color), SNAME("L
    matches_label->set_text(vformat(TTR("%d replaced."), rc));

    callable_mp((Object *)text_editor, &Object::connect).call_deferred(SceneStringName(text_changed), callable_mp(this, &FindReplaceBar
    results_count = -1;
    results_count_to_current = -1;
    needs_to_count_results = true;
}

void FindReplaceBar::_get_search_from(int &r_line, int &r_col, SearchMode p_search_mode) {
    if (!text_editor->has_selection(0) || is_selection_only()) {
        r_line = text_editor->get_caret_line(0);
        r_col = text_editor->get_caret_column(0);

        if (p_search_mode == SEARCH_PREV && r_line == result_line && r_col >= result_col && r_col <= result_col + get_search_text().len
            r_col = result_col;
        }
        return;
    }

    if (p_search_mode == SEARCH_NEXT) {
```

PROBLEMS 293 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Filter (e.g. text, **/*.ts, !**/node_modules/**)

code_editor.cpp godot-debug-assignment\editor 256

```
Limiting analysis of branches. Use --check-level=exhaustive to analyze all branches. CppCheck (c-cpp-flylint)(normalCheckLevelMaxBranches) [Ln 1, Col 1]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 349, Col 2]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 566, Col 3]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 569, Col 3]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_ca... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 1342, Col 2]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_ca... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 1351, Col 2]
```



PROBLEMS 293 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Filter (e.g. text, **/*.ts, !**/node_modules/**)

code_editor.cpp godot-debug-assignment\editor 256

```
Limiting analysis of branches. Use --check-level=exhaustive to analyze all branches. CppCheck (c-cpp-flylint)(normalCheckLevelMaxBranches) [Ln 1, Col 1]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 349, Col 2]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 566, Col 3]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_cas... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 569, Col 3]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_ca... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 1342, Col 2]
C-style pointer casting detected. C++ offers four different kinds of casts as replacements: static_cast, const_cast, dynamic_cast and reinterpret_ca... CppCheck (c-cpp-flylint)(cstyleCast) [Ln 1351, Col 2]
```