Software Engineering Testing Lab

**Name:** Harshit Kapadia

**Id:** 202201486

**Code Inspection:**

Below is a breakdown of issues discovered during the code inspection, categorized by type.

**First 200 Lines Analysis**

**Category A: Data Reference Issues**

* **Uninitialized Variables**:
  + Variables such as name, gender, age, and phone\_no are declared but might be used before being properly assigned values, which could lead to runtime errors.
* **Array Boundaries**:
  + Arrays like char specialization[100] and char name[100] do not implement explicit bounds checking, making them vulnerable to buffer overflow attacks.

**Category B: Data Declaration Issues**

* **Implicit Declarations**:
  + Variables like adhaar and identification\_id should be clearly declared and initialized with appropriate data types to prevent errors during usage.
* **Array Initialization**:
  + String arrays such as char specialization[100] and char gender[100] could benefit from default initialization to avoid working with undefined or garbage values.

**Category C: Computation Errors**

* **Inconsistent Data Handling in Computations**:
  + Since variables like phone\_no and adhaar represent numeric strings, ensure that they are consistently treated as strings to avoid unexpected behavior when performing operations or validations.

**Category E: Control-Flow Issues**

* **Risky Use of goto Statements**:
  + Using goto (e.g., goto C;) in validation routines for Aadhaar and mobile numbers can lead to endless loops or unpredictable control flow. It is recommended to use while loops with clear termination conditions for improved code structure.

**Category F: Interface Issues**

* **Parameter Mismatch**:
  + Functions such as add\_doctor() and display\_doctor\_data() need to ensure that the number and types of parameters passed align correctly with their definitions to prevent interface errors.

**Category G: Input/Output Issues**

* **File Handling Issues**:
  + Operations on files (e.g., Doctor\_Data.data) should ensure files are successfully opened and always closed after use to avoid issues like memory leaks. Additionally, exceptions for failed file operations (e.g., file not found) should be handled explicitly to prevent program crashes.

**Second 200 Lines Analysis**

**Category A: Data Reference Issues**

* **File Handling Deficiencies**:
  + Files such as Doctor\_Data.dat and Patient\_Data.dat are frequently used without proper error handling for potential issues (e.g., file not found). It is essential to handle such exceptions to maintain system reliability.

**Category B: Data Declaration Issues**

* **Array Size Limitations**:
  + Arrays like name[100], specialization[100], and gender[10] may cause buffer overflows if user inputs exceed the predefined length. Consider implementing input length checks to prevent overflow errors.

**Category C: Computation Errors**

* **Vaccine Stock Calculation Vulnerabilities**:
  + The display\_vaccine\_stock() function calculates the total vaccine stock across multiple centers but lacks safeguards against negative inputs or integer overflows. Adding these checks would prevent potential miscalculations.

**Category E: Control-Flow Issues**

* **Overuse of goto Statements**:
  + In functions like add\_doctor() and add\_patient\_data(), multiple goto statements are used to revalidate inputs. These could be replaced with structured loop constructs like while or do-while to improve readability and reduce complexity.

**Category F: Interface Issues**

* **Faulty String Comparisons**:
  + In search\_doctor\_data(), string comparisons involving identification\_id and sidentification\_idrely on .compare(). Ensure all string comparisons are carefully managed to avoid logical errors and inconsistencies.

**Category G: Input/Output Issues**

* **Missing File Closure**:
  + Files opened in search\_center() and display\_vaccine\_stock() should always be closed after use to avoid resource leaks or file locking issues.

**Control Flow Suggestion**

Replacing goto statements with well-structured loops in the input validation sections will improve maintainability and reduce the chances of hard-to-diagnose bugs. Loops with clear exit conditions ensure smoother flow control and more readable code.

**Third 200 Lines Inspection**

**Category A: Data Reference Issues**

* **File Handling Validation**:
  + In functions like add\_vaccine\_stock() and display\_vaccine\_stock(), ensure that files for vaccine centers (e.g., center1.txt, center2.txt) are successfully opened before proceeding with operations. Incorporate error checking to catch any file access failures.

**Category B: Data Declaration Issues**

* **Inconsistent Data Types**:
  + The variables adhaar and phone\_no are expected to hold numeric string values but are treated inconsistently across multiple functions. Ensure uniform handling of these values to avoid misinterpretation as integers.

**Category C: Computation Errors**

* **Vaccine Stock Calculation**:
  + In display\_vaccine\_stock(), incorrect stock totals may occur if values are uninitialized or negative. Add safeguards to initialize all relevant variables and validate the stock numbers before summation.

**Category E: Control-Flow Issues**

* **Overuse of goto Statements**:
  + The use of goto in functions like search\_doctor\_data() and add\_doctor() complicates the control flow. Adopting structured loops such as while or for would make the code easier to understand and maintain.

**Category F: Interface Issues**

* **Parameter Consistency**:
  + Verify that functions like search\_by\_aadhar() maintain consistent parameter expectations across all modules to prevent mismatches and runtime errors.

**Category G: Input/Output Issues**

* **Improper File Closing**:
  + Files like Doctor\_Data.data are sometimes left open in certain code paths. Make sure every file operation ends with proper closing to avoid resource exhaustion or file lock issues.

**Fourth 200 Lines Inspection**

**Category A: Data Reference Issues**

* **Uninitialized Variables**:
  + Functions such as update\_patient\_data(), show\_patient\_data(), and applied\_vaccine() should initialize variables like maadhaar and file streams explicitly to prevent unexpected behavior due to uninitialized data.

**Category B: Data Declaration Issues**

* **Array Length Limits**:
  + Character arrays like gender[10] and adhaar[12] risk overflow if input values exceed the defined size. Input validation is necessary to mitigate this.

**Category C: Computation Errors**

* **Incorrect Dose Incrementation**:
  + In update\_patient\_data(), using dose++ directly could lead to invalid counts without checks. Validate the dose range before incrementing to ensure data accuracy.

**Category E: Control-Flow Issues**

* **Excessive Reliance on goto**:
  + The use of goto in search\_doctor\_data() and add\_patient\_data() leads to tangled logic. Replacing these with loops would streamline the code structure and enhance readability.

**Category F: Interface Issues**

* **String Comparison Errors**:
  + Functions like search\_by\_aadhar() compare string variables (e.g., adhaar.compare(sadhaar)) but may not account for edge cases. Implement consistent string handling to prevent errors.

**Category G: Input/Output Issues**

* **File Handling Vulnerabilities**:
  + Files like Patient\_Data.dat and Doctor\_Data.dat are opened in several functions without adequate error handling. Ensure proper exception handling to avoid runtime crashes.

**Fifth 200 Lines Inspection**

**Category A: Data Reference Issues**

* **Uninitialized Variables**:
  + Variables such as maadhaar in update\_patient\_data() and other fields in search\_doctor\_data() should be initialized to prevent usage before assignment.

**Category B: Data Declaration Issues**

* **Array Boundary Issues**:
  + Arrays such as gender[10] risk buffer overflows if input exceeds the allocated space. Validate input lengths to avoid this issue.

**Category C: Computation Errors**

* **Unchecked Dose Increment**:
  + In update\_patient\_data(), the dose++ operation is performed without validation, which can result in invalid dose counts. Add proper checks to avoid such errors.

**Category E: Control-Flow Issues**

* **Redundant Use of goto**:
  + Functions like search\_doctor\_data() and add\_doctor() contain multiple goto statements, making the logic hard to follow. Consider using loops for better maintainability.

**Category F: Interface Issues**

* **Parameter Handling Issues**:
  + Ensure that parameters, such as those in search\_by\_aadhar(), are passed correctly with consistent types across all modules.

**Category G: Input/Output Issues**

* **Unclosed File Operations**:
  + Some file operations in Patient\_Data.dat and Doctor\_Data.data leave files open in specific code paths. Always close files after use to avoid resource leaks.

**Final 300 Lines Inspection**

**Category A: Data Reference Issues**

* **File Access without Error Handling**:
  + Files like center1.txt, center2.txt, and center3.txt are accessed in add\_vaccine\_stock() and display\_vaccine\_stock() without adequate error handling. Add checks to handle file access errors gracefully.

**Category B: Data Declaration Issues**

* **Uninitialized Variables**:
  + Variables like sum\_vaccine\_c1, sum\_vaccine\_c2, and sum\_vaccine\_c3 should be initialized to prevent undefined behavior during vaccine stock calculations.

**Category C: Computation Errors**

* **Invalid Stock Values**:
  + In add\_vaccine\_stock(), ensure that stock values are always positive and valid to prevent incorrect calculations when displayed.

**Category E: Control-Flow Issues**

* **Overuse of goto Statements**:
  + In functions like add\_doctor() and add\_patient\_data(), the frequent use of goto makes the code challenging to maintain. Loops would offer a cleaner and more structured approach.

**Category G: Input/Output Issues**

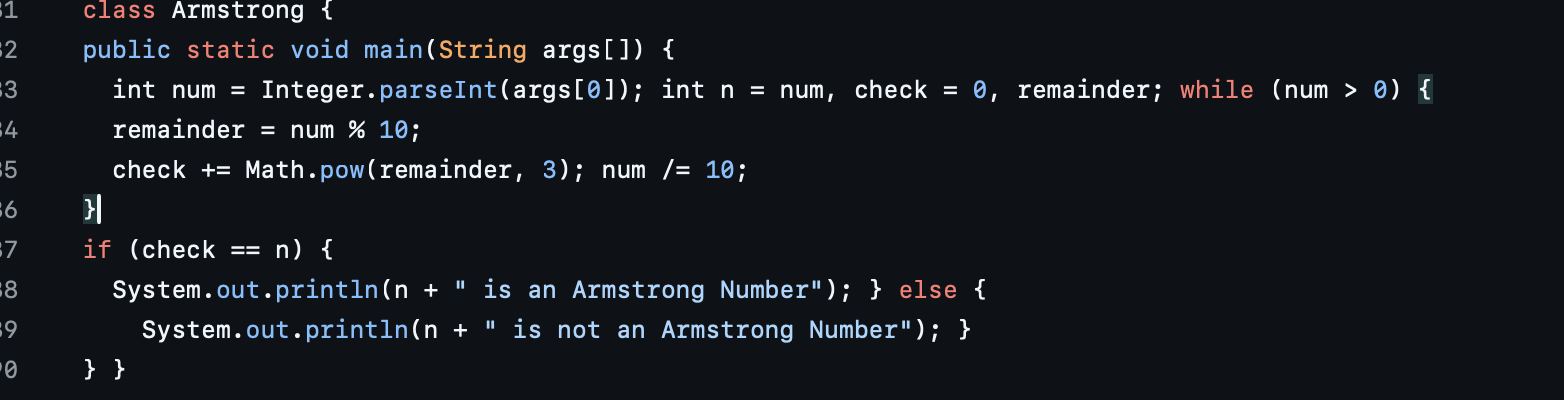
* **Inconsistent File Closure**:
  + In some file-handling branches, files are not closed correctly, risking memory leaks. Ensure every opened file is closed after operations and handle exceptions where necessary.

### ****DEBUGGING****

### ****1. Armstrong Number Program****

* **Error:** Incorrect computation of the remainder.
* **Fix:** Use breakpoints to validate remainder calculation logic.

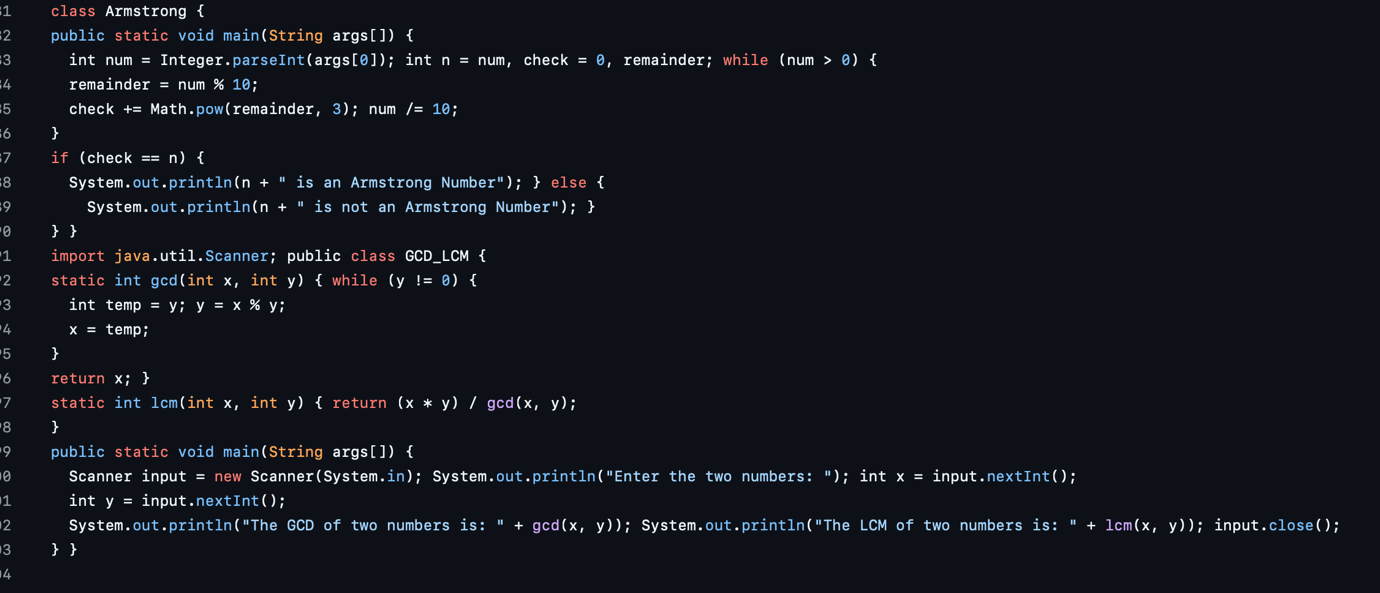
#### **Corrected Code:**



### ****2. GCD and LCM Program****

* **Errors:**
  1. Incorrect while loop condition in GCD.
  2. Incorrect LCM logic.
* **Fix:** Add breakpoints to validate GCD and LCM operations.

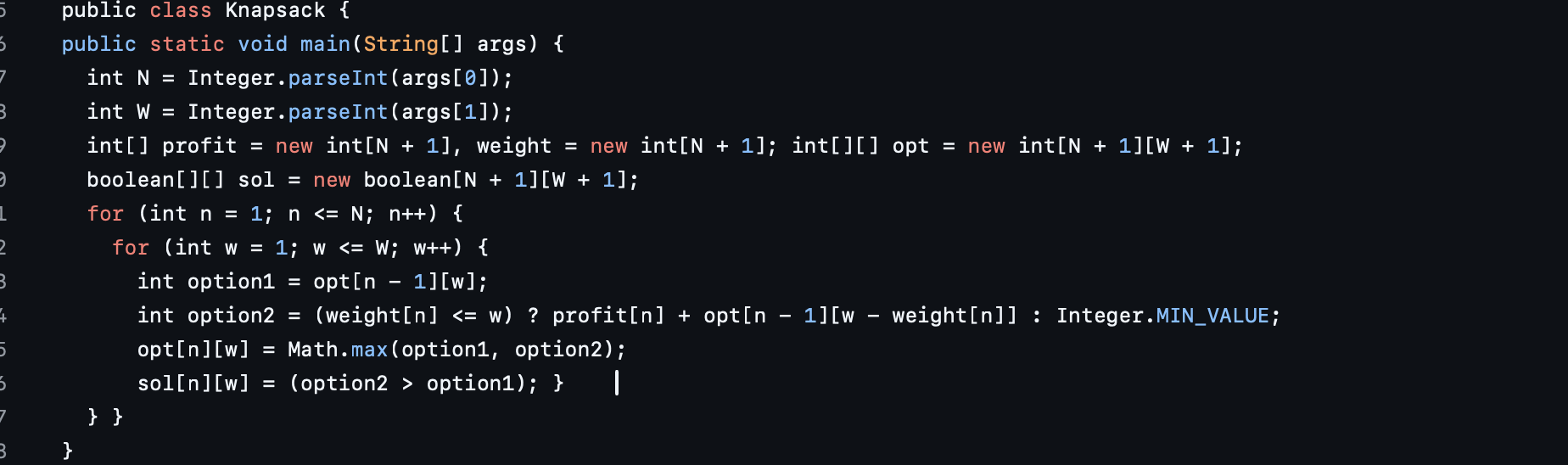
#### **Corrected Code:**



### ****3. Knapsack Program****

* **Error:** Incorrect n increment in the loop.
* **Fix:** Add breakpoints to check loop behavior.

#### **Corrected Code:**



**4. Magic Number Program**

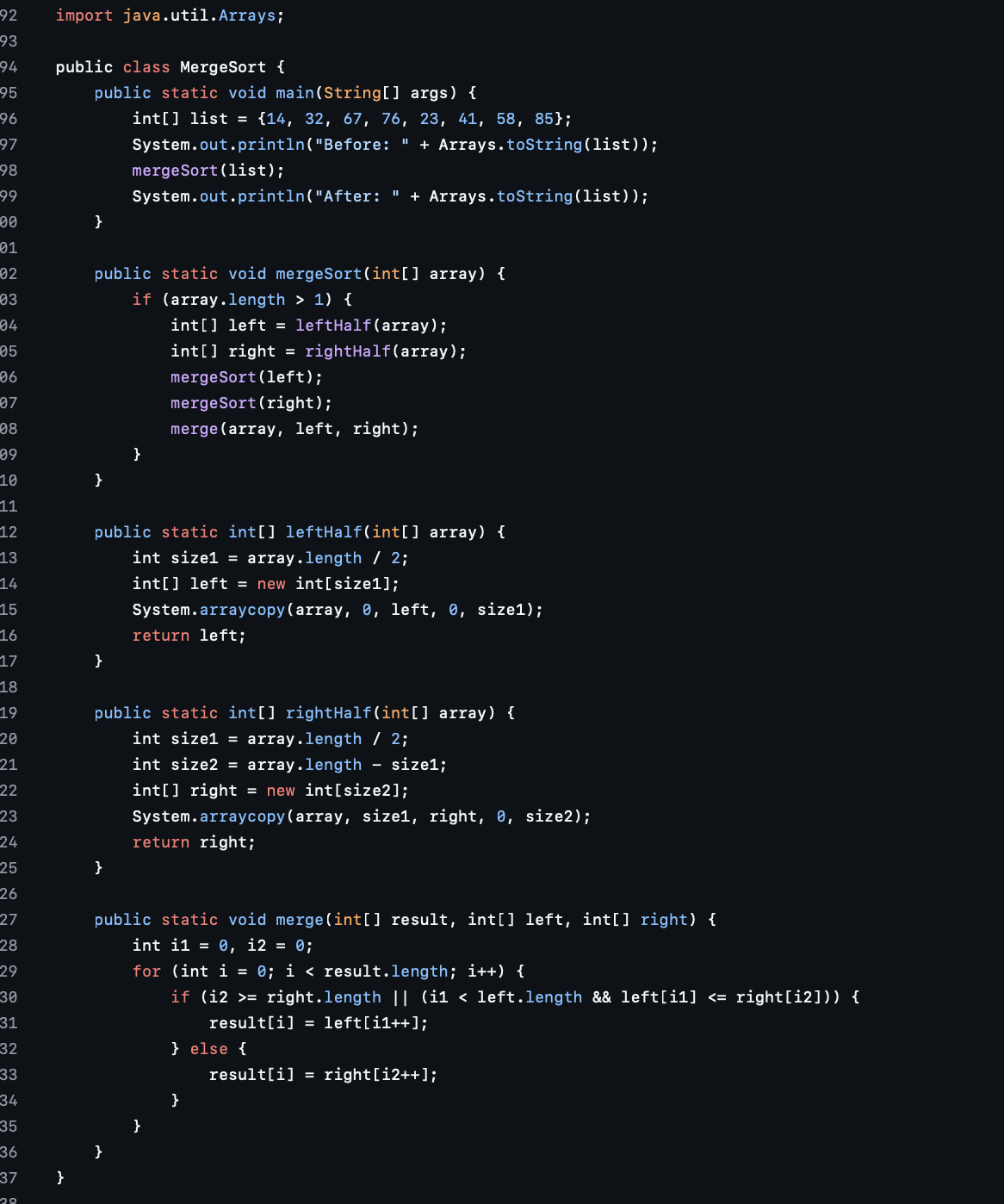
* **Errors:**
  1. Incorrect condition in inner while loop.
  2. Missing semicolons.
* **Fix:** Validate while loop logic with breakpoints.



### ****5. Merge Sort Program****

* **Errors:**
  1. Incorrect array splitting logic.
  2. Incorrect merge operation inputs.
* **Fix:** Add breakpoints to debug merging and splitting.

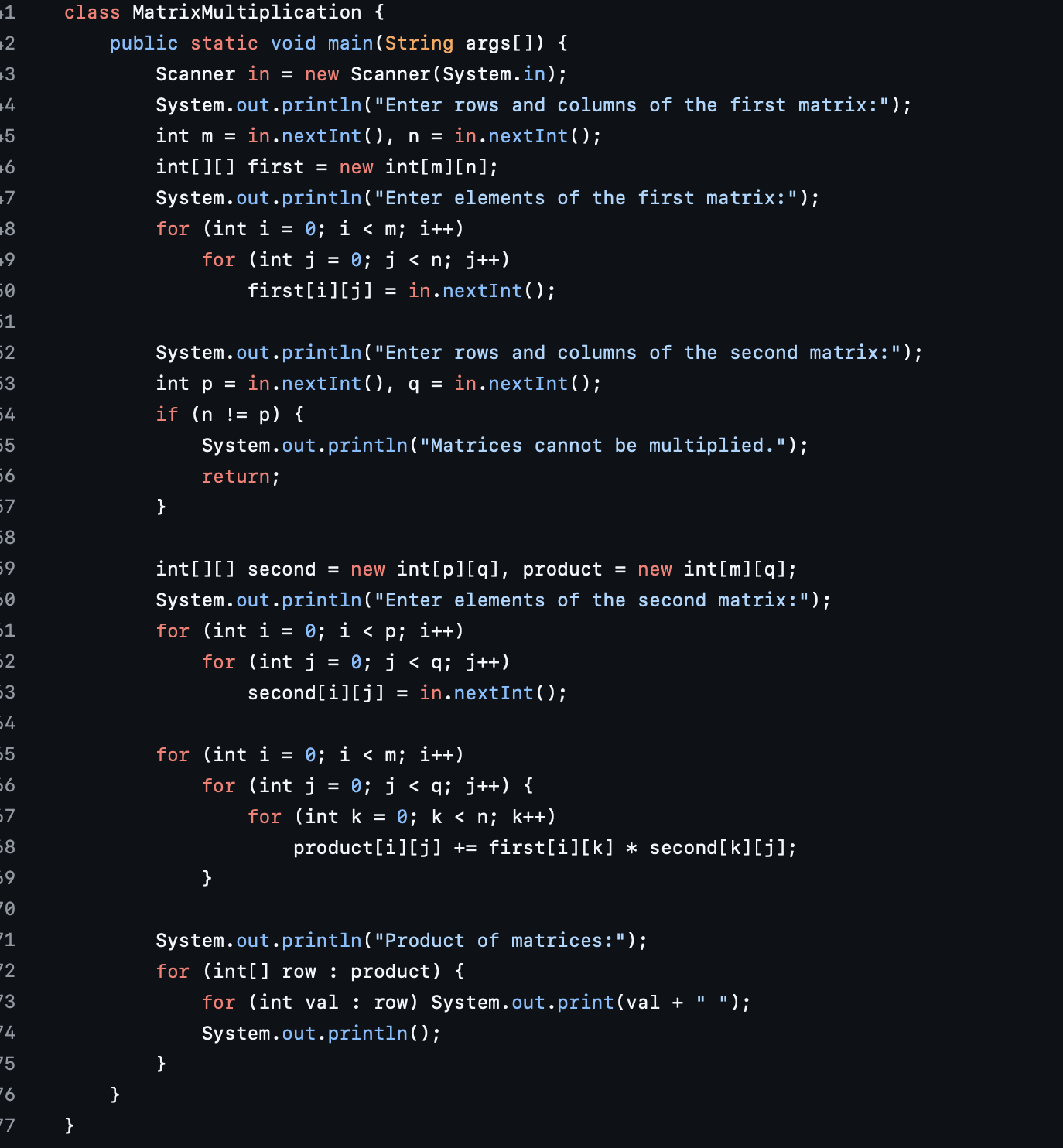
#### **Corrected Code:**



### ****6. Multiply Matrices Program****

* **Errors:**
  1. Incorrect loop indices.
  2. Wrong error message.
* **Fix:** Debug matrix multiplication logic with breakpoints.

#### **Corrected Code:**



**7. Quadratic Probing Hash Table Program**

**Errors:**

1. **Typos** in the insert, remove, and get methods.
2. **Incorrect rehashing logic.**

**Corrected Code:**

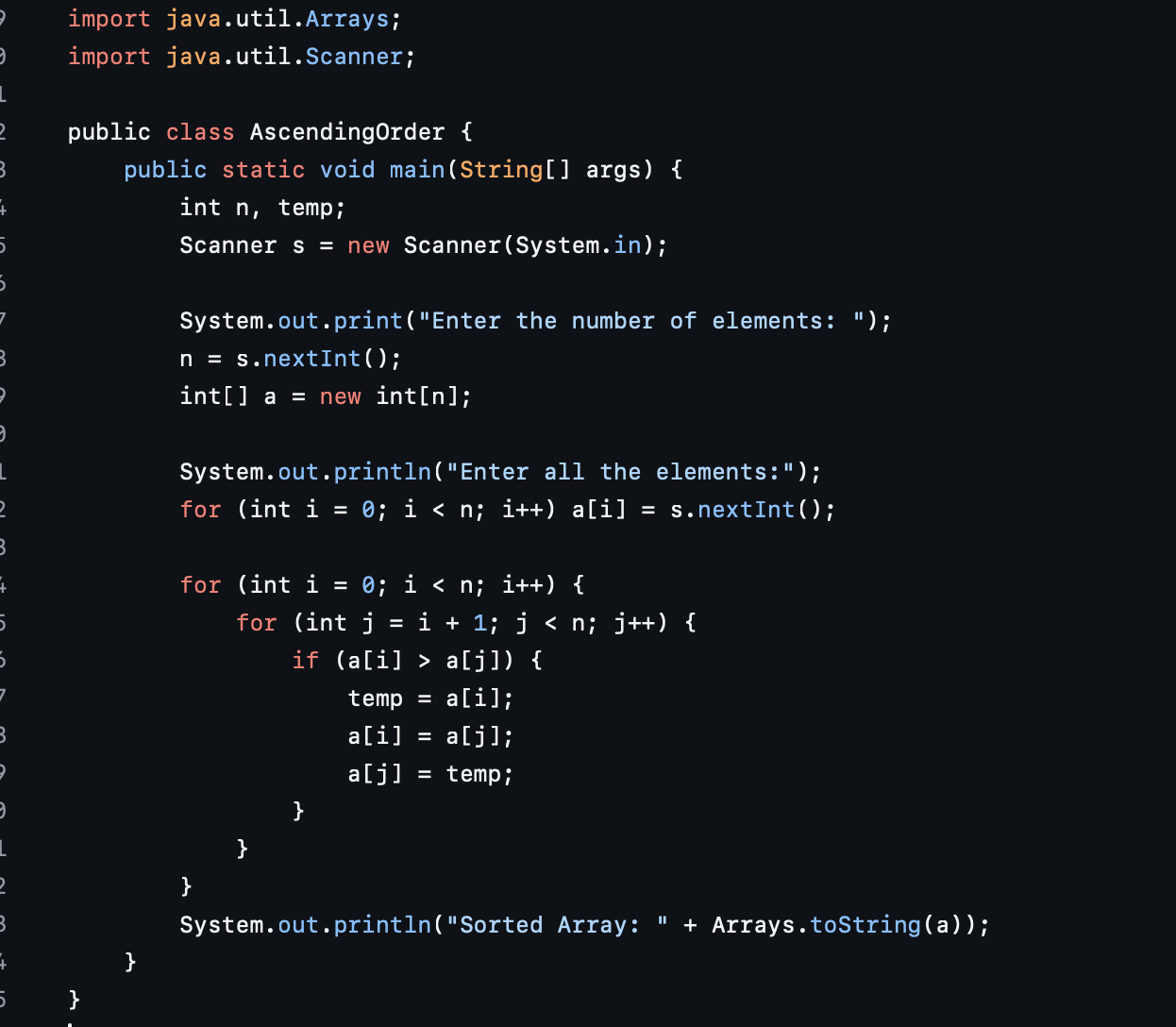


**8. Sorting Array Program**

**Errors:**

1. **Incorrect class name** with an extra space.
2. **Incorrect loop condition** and extra semicolon.

**Corrected Code:**

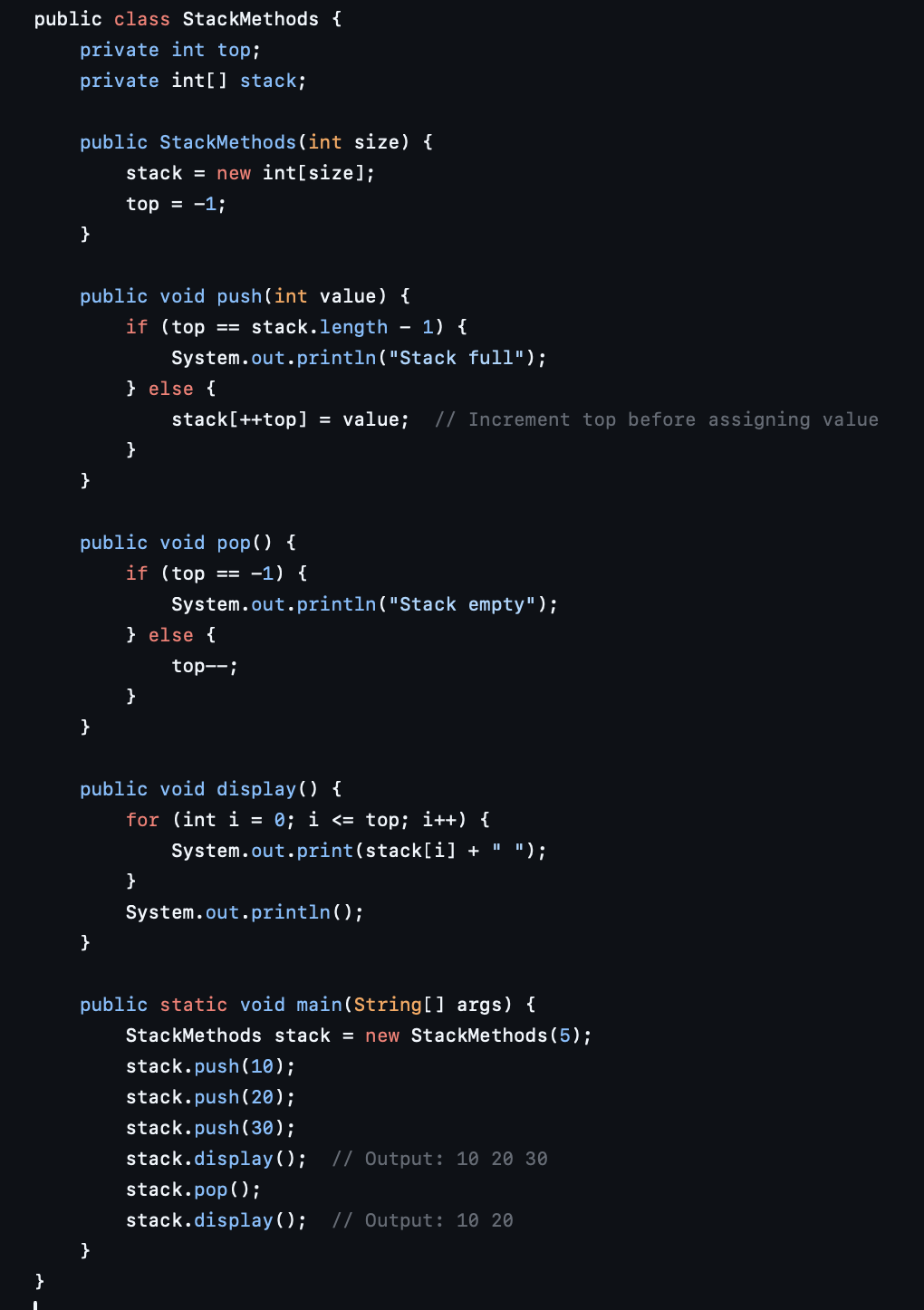
****

**9. Stack Implementation Program**

**Errors:**

1. **Incorrect top increment** (top-- instead of top++ in push).
2. **Incorrect loop condition** in display.
3. **Missing pop method.**

**Corrected Code:**



**10. Tower of Hanoi Program**

**Error:**

1. **Incorrect increment/decrement** in recursive calls.



**STATIC ANALYSIS TOOL:**

**Program inspected:** 1300 lines of code  
**Analysis Tool:** Cppcheck

### ****Results and Warnings:****

**Information:**  
Cppcheck reported the following standard headers were **not found**, but these are **not required** for accurate static analysis:

<stdio.h>, <stdlib.h>,<sys/types.h>,<sys/stat.h>, <unistd.h>, <dirent.h>, <fcntl.h>, <libgen.h>, <errno.h>, <string.h>, <windows.h>, <iostream>, <fstream>, <conio.h>, <iomanip>, <cstdlib>, <string>

**[202201486\_Lab3\_2.c]:**

* [Line 0]: (information) Limiting analysis of branches. Use --check-level=exhaustive for complete branch coverage.
* [Line 116, 120, 126, 127, 133]: (warning) **scanf()** used without field width limits; this can crash with large inputs.
* [Line 34]: (style) The scope of variable **'ch'** can be reduced.
* [Line 115]: (style) The scope of variable **'path2'** can be reduced.
* [Line 16]: (style) Parameter **'file'** can be declared as a pointer to const.
* [Line 55]: (style) Variable **'direntp'** can be declared as a pointer to const.
* [Line 40]: (warning) **fgetc()** return value stored in char variable and compared with EOF, which can cause issues.

**[202201486\_Lab3\_3.c]:**

* [Lines 1-5]: (information) Missing standard headers detected: <stdio.h>, <stdlib.h>, <sys/types.h>, <sys/stat.h>, <unistd.h>.

**[202201486\_Lab3\_1.c]:**

* [Lines 1-9]: (information) Same missing headers as above.
* [Line 29]: (style) The scope of variable **'ch'** can be reduced.
* [Line 11]: (style) Parameter **'file'** can be declared as a pointer to const.
* [Line 50]: (style) Variable **'direntp'** can be declared as a pointer to const.
* [Line 35]: (warning) **fgetc()** return value stored in char and compared with EOF, which can cause errors.

**[Covid-Management-System.cpp]:**

* [Lines 4-12]: (information) Missing headers include <iostream>, <cstring>, <windows.h>, <fstream>, <conio.h>, <iomanip>, <cstdlib>, <string>, <unistd.h>.
* [Line 562, 565, 614, 1121]: (portability) Calling **fflush()** on stdin may cause undefined behavior on non-Linux systems.
* [Lines 538-1317]: (style) Multiple instances of **C-style pointer casting** detected.
* [Line 427, 443, 459, 892]: (style) Unnecessary consecutive **return, break, continue, goto**, or **throw**statements.
* [Line 306]: (style) The scope of variable **'usern'** can be reduced.
* [Line 48 -> 277]: (style) Local variable **'user'** shadows an outer function.
* [Line 40 -> 304]: (style) Local variable **'c'** shadows an outer variable.
* [Line 275]: (performance) Function parameter **'str'** should be passed as a **const reference**.
* [Line 277, 304]: (style) **Unused variables**: 'user', 'c'.