Lab 7:

Code inspection, Debugging and Static analysis tool

Jaimin Prajapati - 202201228

20th October, 2024

C language 2000 line of code link :

https://github.com/202201228/IT314_Software_Engineering/blob/main/C_language_2000LOC.c

Program Inspection: (Submit the answers of following questions for each code fragment)

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

Answer :

There are no errors identified that would cause the program to fail or crash. However, numerous style issues are present. These are not critical errors .Based on the analysis, here are the types of issues identified:

- **Variable Scope Reduction:** Several variables are flagged for having too broad a scope (`ch`, `tmq`, `rio`, `stio`, `endblk`, etc.). This affects code efficiency and readability but isn't a runtime error.
- **Shadowed Variables:** The variable `i` is shadowed by another variable. This can lead to confusion or incorrect results if not handled carefully.
- **Pointer to Const Recommendations:** Many variables and parameters can be declared as pointers to `const` (e.g., `p`, `c`, `d`), improving safety by preventing accidental modification of data.
- **Unused Struct Members:** Several struct members (like `TYPEUIDCWB:sqlCode`, `TYPEDBDIN:only`) and union members (`TYPEPAGE64K:buf`) are flagged as never used. This can clutter the code and should be reviewed.

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

Answer : Category A: Data Reference Errors and Category B: Data-Handling Errors are quite effective.Because issues like variable scope reduction, which relates to memory management and efficiency, and shadowed variables, align well with data-handling and data reference problems. These categories focus on ensuring that the variables are appropriately managed and accessed.

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

Answer :

This inspection didn't identify runtime errors or logical errors. It focuses on style and potential data misuse but does not detect issues such as:

- **Logic errors:** If the algorithm or logic implemented in the code is incorrect, it will not give error it. For example, incorrect mathematical operations or incorrect handling of data structures would not be detected.
- **Runtime errors:** Issues like memory leaks, buffer overflows, segmentation faults, or division by zero would not be detected by static analysis. These would require runtime checks or dynamic analysis tools like Valgrind or manual debugging.
- **Performance bottlenecks:** it does not analyze the runtime performance of code.

4. Is the program inspection technique is worth applicable?

Answer : Yes, the program inspection technique is worth applying. It provides valuable insights into the quality of code, identifying potential bugs and inefficiencies early in the development process. Although it focuses more on coding style, variable usage, and optimizations, it helps in maintaining clean, robust, and maintainable code, reducing future errors.

Debugging: (Submit the answers of following questions for each code fragment)

1. ARMSTRONG

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

- Logic Error 1: The remainder calculation is incorrect. Instead of dividing `num` by 10 (`remainder = num / 10;`), we should be using the modulus operator `%` to get the remainder.
- Logic Error 2: The number update is incorrect. Instead of updating `num = num % 10;`, we should be dividing the number by 10 to eliminate the last digit (`num = num / 10;`).

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: On the line `remainder = num / 10;` to identify the incorrect calculation of the remainder.
- Breakpoint 2: On the line `num = num % 10;` to identify the incorrect logic for updating the number.

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

- Change the remainder calculation from `remainder = num / 10;` to `remainder = num % 10;`.
- Update the number by dividing `num` by 10 instead of using modulus: change `num = num % 10;` to `num = num / 10;`.

➢ **Submit your complete executable code?**

```
class Armstrong {
    public static void main(String args[]) {
        int num = Integer.parseInt(args[0]);
        int n = num;
        int check = 0, remainder;

        while (num > 0) {
            remainder = num % 10;
            check = check + (int) Math.pow(remainder, 3);
            num = num / 10;
        }

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

2. GCD AND LCM

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

- Error 1: In the `gcd` method, the condition `while(a % b == 0)` is incorrect. It should be `while(a % b != 0)` to properly calculate the GCD.
- Error 2: In the `lcm` method, the condition `if(a % x != 0 && a % y != 0)` is incorrect. It should be `if(a % x == 0 && a % y == 0)` to find the correct least common multiple.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the `gcd` method on the condition `while(a % b == 0)` to correct it.
- Breakpoint 2: In the `lcm` method on the condition `if(a % x != 0 && a % y != 0)` to correct it.

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

- Update the `while(a % b == 0)` condition in the `gcd` method to `while(a % b != 0)`.
- Update the condition `if(a % x != 0 && a % y != 0)` in the `lcm` method to `if(a % x == 0 && a % y == 0)`.

➢ **Submit your complete executable code?**

```
import java.util.Scanner;

public class GCD_LCM

{

  static int gcd(int x, int y)

  {
    int r=0, a, b;

    a = (x > y) ? y : x;

    b = (x < y) ? x : y;


    r = b;

    while(a % b != 0)

    {

        r = a % b;

        a = b;

        b = r;

    }

    return r;

}


static int lcm(int x, int y)

{

    int a;

    a = (x > y) ? x : y;

    while(true)

    {

        if(a % x == 0 && a % y == 0)

            return a;
            ++a;
        }
    }

    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the two numbers: ");
        int x = input.nextInt();
        int y = input.nextInt();

        System.out.println("The GCD of two numbers is: " + gcd(x, y));
        System.out.println("The LCM of two numbers is: " + lcm(x, y));
        input.close();
    }
}
```

3. KNAOPSACK

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

- Error 1: In the line `int option1 = opt[n++][w];`, using `n++` increments `n` unintentionally. It should just be `n`, so the line should be changed to `int option1 = opt[n][w];`.
- Error 2: In the line `if (weight[n] > w) option2 = profit[n-2] + opt[n-1][w-weight[n]];`, `weight[n] > w` should be `weight[n] <= w` to ensure the item can be taken, and the calculation `profit[n-2]` should be changed to `profit[n]` to correctly refer to the profit of the current item.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: On the line `int option1 = opt[n++][w];` to check the increment issue.
- Breakpoint 2: On the condition `if (weight[n] > w)` and the profit calculation to check the correct condition and index.

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

- Replace `int option1 = opt[n++][w];` with `int option1 = opt[n][w];`.
- Change `if (weight[n] > w)` to `if (weight[n] <= w)`.
- Correct `profit[n-2]` to `profit[n]`.

➢ **Submit your complete executable code?**

//Knapsack

public class Knapsack {

public static void main(String[] args) {

```
int N = Integer.parseInt(args[0]);   // number of items

int W = Integer.parseInt(args[1]);   // maximum weight of knapsack


int[] profit = new int[N+1];

int[] weight = new int[N+1];


// generate random instance, items 1..N

for (int n = 1; n <= N; n++) {

    profit[n] = (int) (Math.random() * 1000);

    weight[n] = (int) (Math.random() * W);

}


// opt[n][w] = max profit of packing items 1..n with weight limit w

// sol[n][w] = does opt solution to pack items 1..n with weight limit w include item
n?

int[][] opt = new int[N+1][W+1];

boolean[][] sol = new boolean[N+1][W+1];


for (int n = 1; n <= N; n++) {

    for (int w = 1; w <= W; w++) {


        // don't take item n

        int option1 = opt[n][w]; // Corrected increment issue
```

```
            // take item n
            int option2 = Integer.MIN_VALUE;
            if (weight[n] <= w) option2 = profit[n] + opt[n-1][w-weight[n]]; // Corrected
condition and profit index

            // select better of two options
            opt[n][w] = Math.max(option1, option2);
            sol[n][w] = (option2 > option1);
        }
    }

    // determine which items to take
    boolean[] take = new boolean[N+1];
    for (int n = N, w = W; n > 0; n--) {
        if (sol[n][w]) { take[n] = true;  w = w - weight[n]; }
        else           { take[n] = false;                    }
    }

    // print results
    System.out.println("item" + "\t" + "profit" + "\t" + "weight" + "\t" + "take");
    for (int n = 1; n <= N; n++) {
        System.out.println(n + "\t" + profit[n] + "\t" + weight[n] + "\t" + take[n]);
        }
    }
}
```

4. MAGIC NUMBER

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

- Error 1: In the inner while loop, the condition `while(sum == 0)` is incorrect. It should be `while(sum != 0)` because we need to process the digits of `sum` until it becomes 0.
- Error 2: In the statement `s = s * (sum / 10);`, it should be `s = s + (sum % 10);` to correctly calculate the sum of digits instead of performing multiplication.
- Error 3: Missing semicolon after `sum=sum%10`.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the condition `while(sum == 0)` to check the logic for the sum of digits.
- Breakpoint 2: In the statement `s = s * (sum / 10);` to check the digit summing operation.

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

- Change `while(sum == 0)` to `while(sum != 0)`.
- Change `s = s * (sum / 10);` to `s = s + (sum % 10);` to correctly sum the digits.
- Add the missing semicolon after `sum=sum%10;`.

➢ **Submit your complete executable code?**

```
import java.util.*;

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) // Corrected the condition

            {

                s = s + (sum % 10); // Corrected the sum calculation

                sum = sum / 10; // Fixed missing semicolon

            }

            num = s;
        }


        if(num == 1)

        {

            System.out.println(n + " is a Magic Number.");

        }

        else

        {

            System.out.println(n + " is not a Magic Number.");

        }

    }

}
```

5. MERGE SORT

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

- Error 1: In the line `int[] left = leftHalf(array+1);`, passing `array+1` is incorrect. It should be just `array`, as we need to split the original array. The same issue exists in the line `int[] right = rightHalf(array-1);`.
- Error 2: In the line `merge(array, left++, right--);`, `left++` and `right--` are incorrect. We should simply pass `left` and `right` arrays without incrementing or decrementing.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the lines where array splitting occurs (`leftHalf` and `rightHalf`) to check the input.
- Breakpoint 2: In the merge function to verify if the correct arrays are being merged.

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

- Change `int[] left = leftHalf(array+1);` to `int[] left = leftHalf(array);`.
- Change `int[] right = rightHalf(array-1);` to `int[] right = rightHalf(array);`.
- Change `merge(array, left++, right--);` to `merge(array, left, right);`.

➢ **Submit your complete executable code?**

```
import java.util.*;


public class MergeSort {

   public static void main(String[] args) {
    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);  // Fixed incorrect array splitting

        int[] right = rightHalf(array);  // Fixed incorrect array splitting


        // recursively sort the two halves

        mergeSort(left);

        mergeSort(right);


        // merge the sorted halves into a sorted whole

        merge(array, left, right);  // Fixed incorrect incrementing/decrementing

    }
```

```
}

// 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;
}
```

```
  // 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++;
          }
      }
  }
}
```

6. MULTIPLY MATRICS

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

- Error 1: In the matrix multiplication logic, accessing elements using `first[c-1][c-k]` and `second[k-1][k-d]` is incorrect. Indices must be used directly without decrementing them.
- Error 2: The nested loops for matrix multiplication are not using the correct index bounds. Specifically, the `k` loop should iterate up to `n` (number of columns of the first matrix or number of rows of the second matrix), not `p` (the number of rows of the second matrix).
-

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the section where matrix elements are accessed (during multiplication).
- Breakpoint 2: After the multiplication loop to check the final matrix values.

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

- Replace `first[c-1][c-k]` with `first[c][k]` and `second[k-1][k-d]` with `second[k][d]`.
- Fix the loop variable `k` to iterate up to `n` instead of `p`.

➢ **Submit your complete executable code?**

import java.util.Scanner;

```
class MatrixMultiplication

{

  public static void main(String args[])

  {

    int m, n, p, q, sum = 0, c, d, k;


    Scanner in = new Scanner(System.in);

    System.out.println("Enter the number of rows and columns of first matrix");

    m = in.nextInt();

    n = in.nextInt();


    int first[][] = new int[m][n];


    System.out.println("Enter the elements of first matrix");


    for ( c = 0 ; c < m ; c++ )

      for ( d = 0 ; d < n ; d++ )

        first[c][d] = in.nextInt();


    System.out.println("Enter the number of rows and columns of second matrix");

    p = in.nextInt();

    q = in.nextInt();
    if ( n != p )
      System.out.println("Matrices with entered orders can't be multiplied with each
other.");
    else
    {
      int second[][] = new int[p][q];
      int multiply[][] = new int[m][q];

      System.out.println("Enter the elements of second matrix");

      for ( c = 0 ; c < p ; c++ )
        for ( d = 0 ; d < q ; d++ )
          second[c][d] = in.nextInt();

      for ( c = 0 ; c < m ; c++ )
      {
        for ( d = 0 ; d < q ; d++ )
        {
          for ( k = 0 ; k < n ; k++ )  // Fix: iterate up to n, not p
          {
            sum = sum + first[c][k] * second[k][d];  // Fix: correct element access
          }
```

```
            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");

        }

      }

    }

  }
```

7. QUARDATIC PROBING

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

- Error 1: In the `insert` method, the line `i + = (i + h / h--) % maxSize;` has incorrect syntax. It should be `i = (i + h * h) % maxSize;`, and `h` should be incremented after use.
- Error 2: In the `get` method, the line `i = (i + h * h++) % maxSize;` incorrectly uses `h++`, which does not compute correctly for the next probing. It should be `i = (i + h * h) % maxSize;` and `h` should be incremented after its use.
- Error 3: Similar to the above, the same issue exists in the `remove` method when rehashing elements. It should also use `h * h` instead of `h++`.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the `insert` method where elements are added to the hash table.
- Breakpoint 2: In the `get` method when searching for a value.
- Breakpoint 3: In the `remove` method while rehashing keys.

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

- Updating the calculation of index `i` in `insert`, `get`, and `remove` methods to correctly compute the next index.
- Ensuring `h` is incremented correctly after being used in the index calculation.

➢ **Submit your complete executable code?**

import java.util.Scanner;

```
class QuadraticProbingHashTable

{

    private int currentSize, maxSize;

    private String[] keys;

    private String[] vals;



    public QuadraticProbingHashTable(int capacity)

    {

        currentSize = 0;

        maxSize = capacity;

        keys = new String[maxSize];

        vals = new String[maxSize];

    }



    public void makeEmpty()

    {

        currentSize = 0;

        keys = new String[maxSize];

        vals = new String[maxSize];

    }



    public int getSize()
```

```
{
    return currentSize;
}

public boolean isFull()
{
    return currentSize == maxSize;
}

public boolean isEmpty()
{
    return getSize() == 0;
}

public boolean contains(String key)
{
    return get(key) !=  null;
}

private int hash(String key)
{
    return (key.hashCode() % maxSize + maxSize) % maxSize;
```

```
}


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;
            currentSize++;
            return;
        }
        if (keys[i].equals(key))
        {
            vals[i] = val;
            return;
        }
        i = (tmp + h * h) % maxSize;  // Fix: Correct calculation of index
        h++;  // Fix: Increment h
    } while (i != tmp);
}


public String get(String key)
{
    int i = hash(key), h = 1;
    while (keys[i] != null)
    {
        if (keys[i].equals(key))
            return vals[i];
        i = (i + h * h) % maxSize;  // Fix: Correct calculation of index
        h++;  // Fix: Increment h
    }
    return null;
}


public void remove(String key)
{
    if (!contains(key))
        return;


    int i = hash(key), h = 1;
```

```
        while (!key.equals(keys[i]))
            i = (i + h * h) % maxSize;

        keys[i] = vals[i] = null;

        for (i = (i + h * h) % maxSize; keys[i] != null; i = (i + h * h) % maxSize)
        {
            String tmp1 = keys[i], tmp2 = vals[i];
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
        currentSize--;
    }

    public void printHashTable()
    {
        System.out.println("\nHash Table: ");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] +" "+ vals[i]);
        System.out.println();
    }
}


public class QuadraticProbingHashTableTest
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.println("Enter size");
        QuadraticProbingHashTable qpht = new
QuadraticProbingHashTable(scan.nextInt() );


        char ch;
        do
        {
            System.out.println("\nHash Table Operations\n");
            System.out.println("1. insert ");
            System.out.println("2. remove");
            System.out.println("3. get");
            System.out.println("4. clear");
            System.out.println("5. size");
```

```
int choice = scan.nextInt();

switch (choice)

{

case 1 :

   System.out.println("Enter key and value");

   qpht.insert(scan.next(), scan.next() );

   break;

case 2 :

   System.out.println("Enter key");

   qpht.remove( scan.next() );

   break;

case 3 :

   System.out.println("Enter key");

   System.out.println("Value = "+ qpht.get( scan.next() ));

   break;

case 4 :

   qpht.makeEmpty();

   System.out.println("Hash Table Cleared\n");

   break;

case 5 :

   System.out.println("Size = "+ qpht.getSize() );

   break;
```

```
        default :
            System.out.println("Wrong Entry \n ");
            break;
        }

        qpht.printHashTable();

        System.out.println("\nDo you want to continue (Type y or n) \n");
        ch = scan.next().charAt(0);
    } while (ch == 'Y'|| ch == 'y');
  }
}
```

8. SORTING ARRAY

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

- Class Name Error: The class name contains a space: `public class Ascending _Order` should be `public class AscendingOrder`.
- Loop Condition Error: The loop condition in the first `for` loop for sorting is incorrect. It should be `i < n` instead of `i >= n`, which prevents the loop from executing. The semicolon (`;`) after the first `for` loop also makes it an empty statement, which causes the inner loop to not work as intended.
- Sorting Logic: The condition inside the `if` statement should be `if (a[i] > a[j])` for ascending order, instead of `if (a[i] <= a[j])`.

➢ **How many breakpoints you need to fix those errors?**
   - Breakpoint 1: On the line with the class declaration to fix the class name.
   - Breakpoint 2: On the line with the first `for` loop to correct its condition and remove the semicolon.

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

   - Change the class name from `Ascending _Order` to `AscendingOrder`.
   - Modify the first `for` loop condition from `i >= n` to `i < n` and remove the semicolon after the `for` statement.
   - Change the condition in the `if` statement from `if (a[i] <= a[j])` to `if (a[i] > a[j])`.

➢ **Submit your complete executable code?**

```
// sorting the array in ascending order

import java.util.Scanner;


public class AscendingOrder

{

    public static void main(String[] args)

    {

        int n, temp;

        Scanner s = new Scanner(System.in);

        System.out.print("Enter no. of elements you want in 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();
}
for (int i = 0; i < n; i++) // Fixed loop condition
{
    for (int j = i + 1; j < n; j++)
    {
        if (a[i] > a[j]) // Fixed sorting condition
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
}
System.out.print("Ascending Order:");
for (int i = 0; i < n - 1; i++)
{
    System.out.print(a[i] + ",");
```

```
        }
        System.out.print(a[n - 1]);
    }
}
```

9. STACK IMPLEMENTATION

➢ **How many errors are there in the program? Mention the errors you have identified.**
- Push Logic Error: In the `push` method, the line `top--;` is incorrectly decrementing the `top` index. It should be incremented first before assigning the value to the stack, i.e., `top++` before assigning the value.
- Display Loop Condition Error: In the `display` method, the loop condition `i > top` should be `i <= top` to ensure it displays all elements in the stack.
- Pop Logic Error: In the `pop` method, the line `top++;` should be `top--` to correctly remove the top element from the stack.

➢ **How many breakpoints you need to fix those errors?**
- Breakpoint 1: In the `push` method to fix the logic of incrementing `top`.
- Breakpoint 2: In the `display` method to correct the loop condition.
- Breakpoint 3: In the `pop` method to fix the logic for removing the top element.

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

- **In the `push` method, change `top--;` to `top++;` before assigning the value to the stack.**

- In the `display` method, change the loop condition from `i > top` to `i <= top`.
- In the `pop` method, change `top++;` to `top--;` to correctly remove the top element.

➢ **Submit your complete executable code?**

```
// Stack implementation in java

import java.util.Arrays;


public class StackMethods {

   private int top;

   int size;

   int[] stack;


   public StackMethods(int arraySize) {

      size = arraySize;

      stack = new int[size];

      top = -1;

   }


   public void push(int value) {

      if (top == size - 1) {

         System.out.println("Stack is full, can't push a value");

      } else {
        top++; // Increment top to the next index

        stack[top] = value; // Assign value to the stack

    }

}


public void pop() {

    if (!isEmpty()) {

        top--; // Decrement top to remove the element

    } else {

        System.out.println("Can't pop...stack is empty");

    }

}


public boolean isEmpty() {

    return top == -1;

}


public void display() {

    for (int i = 0; i <= top; i++) { // Change condition to i <= top

        System.out.print(stack[i] + " ");

    }

    System.out.println();
    }
}


public class StackReviseDemo {
    public static void main(String[] args) {
        StackMethods newStack = new StackMethods(5);
        newStack.push(10);
        newStack.push(1);
        newStack.push(50);
        newStack.push(20);
        newStack.push(90);


        newStack.display(); // Output: 10 1 50 20 90
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display(); // Output: 10
    }
}
```

10. <u>TOWER OF HANOI</u>

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

- Incorrect Increment/Decrement Logic: In the recursive call `doTowers(topN ++, inter--, from+1, to+1)`, the use of `++` and `--` is incorrect. `topN` should remain the same in recursive calls, and the parameters `inter` and `from` should not be modified in this manner. This leads to incorrect behavior and will cause a compilation error.
- Missing Recursion for the Last Move: The recursive calls should handle the movement of disks correctly. Specifically, the parameters `inter` and `to` need to be passed without changing their values.
- Base Case Output: The base case correctly prints the movement of disk 1, but the function does not return afterward. While this does not cause an error, it's important to ensure that the base case is followed by an explicit return.

➢ **How many breakpoints you need to fix those errors?**

- Breakpoint 1: In the recursive call to fix the increment/decrement logic.
- Breakpoint 2: To check the recursive logic for moving disks correctly.

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

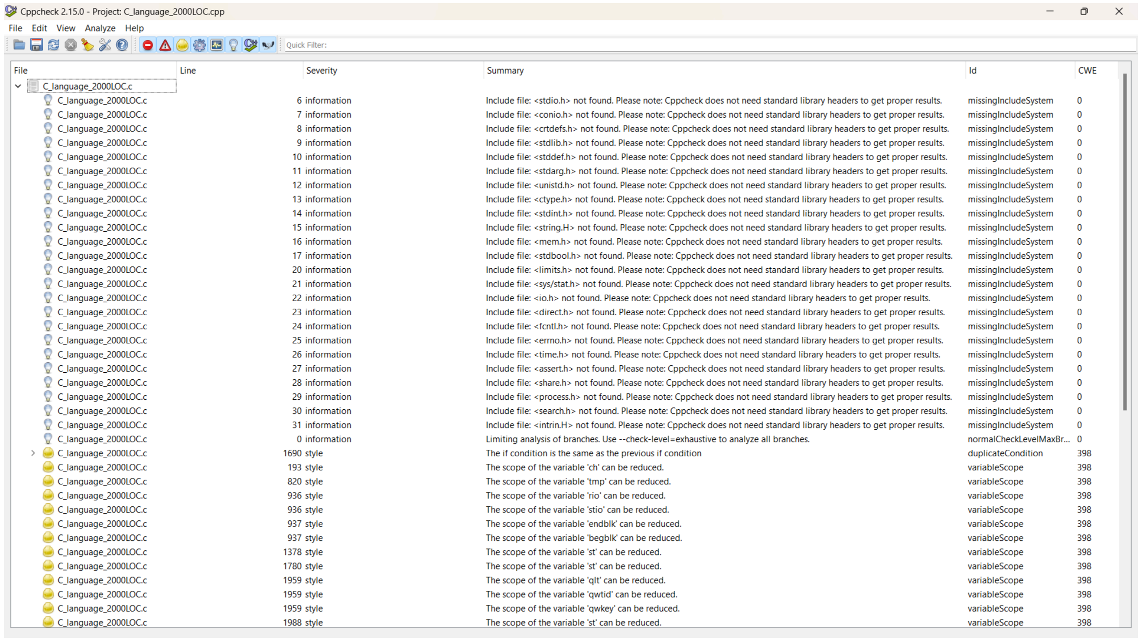
- Change the recursive call from `doTowers(topN ++, inter--, from+1, to+1)` to `doTowers(topN - 1, inter, from, to)`, which correctly passes the current state of the parameters.
- Ensure the parameters are passed correctly in the recursive calls without modifying their values.
- Make sure that all recursive calls have the correct parameters for the Tower of Hanoi algorithm.

➢ **Submit your complete executable code?**

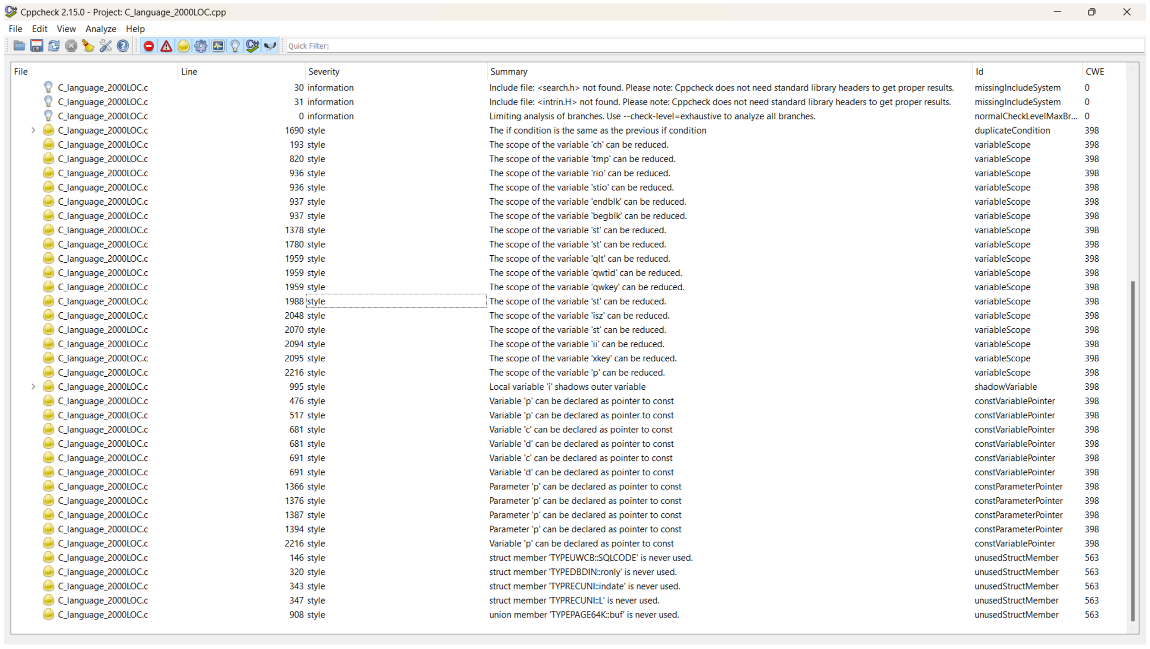
```
public class MainClass {
  public static void main(String[] args) {
    int nDisks = 3;
    doTowers(nDisks, 'A', 'B', 'C');
  }

  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);
      System.out.println("Disk " + topN + " from " + from + " to " + to);
      doTowers(topN - 1, inter, from, to);
    }
  }
}
```

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.


Cppcheck 2.15.0 - Project: C_language_2000LOC.cpp
File Edit View Analyze Help
Quick Filter:

File	Line	Severity	Summary	Id	CWE
C_language_2000LOC.c					
C_language_2000LOC.c	6	information	Include file: <stdio.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	7	information	Include file: <conio.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	8	information	Include file: <crtdefs.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	9	information	Include file: <stdlib.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	10	information	Include file: <stddef.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	11	information	Include file: <stdarg.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	12	information	Include file: <unistd.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	13	information	Include file: <ctype.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	14	information	Include file: <stdint.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	15	information	Include file: <string.H> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	16	information	Include file: <mem.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	17	information	Include file: <stdbool.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	20	information	Include file: <limits.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	21	information	Include file: <sys/stat.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	22	information	Include file: <io.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	23	information	Include file: <direct.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	24	information	Include file: <fcntl.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	25	information	Include file: <errno.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	26	information	Include file: <time.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	27	information	Include file: <assert.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	28	information	Include file: <share.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	29	information	Include file: <process.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	30	information	Include file: <search.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	31	information	Include file: <intrin.H> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	0	information	Limiting analysis of branches. Use --check-level=exhaustive to analyze all branches.	normalCheckLevelMaxBr...	0
C_language_2000LOC.c	1690	style	The if condition is the same as the previous if condition	duplicateCondition	398
C_language_2000LOC.c	193	style	The scope of the variable 'ch' can be reduced.	variableScope	398
C_language_2000LOC.c	820	style	The scope of the variable 'tmp' can be reduced.	variableScope	398
C_language_2000LOC.c	936	style	The scope of the variable 'rio' can be reduced.	variableScope	398
C_language_2000LOC.c	936	style	The scope of the variable 'stio' can be reduced.	variableScope	398
C_language_2000LOC.c	937	style	The scope of the variable 'endblk' can be reduced.	variableScope	398
C_language_2000LOC.c	937	style	The scope of the variable 'begblk' can be reduced.	variableScope	398
C_language_2000LOC.c	1378	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	1780	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qlt' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qwtid' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qwkey' can be reduced.	variableScope	398
C_language_2000LOC.c	1988	style	The scope of the variable 'st' can be reduced.	variableScope	398


Cppcheck 2.15.0 - Project: C_language_2000LOC.cpp
File Edit View Analyze Help
Quick Filter:

File	Line	Severity	Summary	Id	CWE
C_language_2000LOC.c	30	information	Include file: <search.h> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	31	information	Include file: <intrin.H> not found. Please note: Cppcheck does not need standard library headers to get proper results.	missingIncludeSystem	0
C_language_2000LOC.c	0	information	Limiting analysis of branches. Use --check-level=exhaustive to analyze all branches.	normalCheckLevelMaxBr...	0
C_language_2000LOC.c	1690	style	The if condition is the same as the previous if condition	duplicateCondition	398
C_language_2000LOC.c	193	style	The scope of the variable 'ch' can be reduced.	variableScope	398
C_language_2000LOC.c	820	style	The scope of the variable 'tmp' can be reduced.	variableScope	398
C_language_2000LOC.c	936	style	The scope of the variable 'rio' can be reduced.	variableScope	398
C_language_2000LOC.c	936	style	The scope of the variable 'stio' can be reduced.	variableScope	398
C_language_2000LOC.c	937	style	The scope of the variable 'endblk' can be reduced.	variableScope	398
C_language_2000LOC.c	937	style	The scope of the variable 'begblk' can be reduced.	variableScope	398
C_language_2000LOC.c	1378	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	1780	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qlt' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qwtid' can be reduced.	variableScope	398
C_language_2000LOC.c	1959	style	The scope of the variable 'qwkey' can be reduced.	variableScope	398
C_language_2000LOC.c	1988	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	2048	style	The scope of the variable 'isz' can be reduced.	variableScope	398
C_language_2000LOC.c	2070	style	The scope of the variable 'st' can be reduced.	variableScope	398
C_language_2000LOC.c	2094	style	The scope of the variable 'ii' can be reduced.	variableScope	398
C_language_2000LOC.c	2095	style	The scope of the variable 'xkey' can be reduced.	variableScope	398
C_language_2000LOC.c	2216	style	The scope of the variable 'p' can be reduced.	variableScope	398
C_language_2000LOC.c	995	style	Local variable 'i' shadows outer variable	shadowVariable	398
C_language_2000LOC.c	476	style	Variable 'p' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	517	style	Variable 'p' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	681	style	Variable 'c' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	681	style	Variable 'd' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	691	style	Variable 'c' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	691	style	Variable 'd' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	1366	style	Parameter 'p' can be declared as pointer to const	constParameterPointer	398
C_language_2000LOC.c	1376	style	Parameter 'p' can be declared as pointer to const	constParameterPointer	398
C_language_2000LOC.c	1387	style	Parameter 'p' can be declared as pointer to const	constParameterPointer	398
C_language_2000LOC.c	1394	style	Parameter 'p' can be declared as pointer to const	constParameterPointer	398
C_language_2000LOC.c	2216	style	Variable 'p' can be declared as pointer to const	constVariablePointer	398
C_language_2000LOC.c	146	style	struct member 'TYPEUWCB::SQLCODE' is never used.	unusedStructMember	563
C_language_2000LOC.c	320	style	struct member 'TYPEDBDIN::ronly' is never used.	unusedStructMember	563
C_language_2000LOC.c	343	style	struct member 'TYPRECUNI::indate' is never used.	unusedStructMember	563
C_language_2000LOC.c	347	style	struct member 'TYPRECUNI::L' is never used.	unusedStructMember	563
C_language_2000LOC.c	908	style	union member 'TYPEPAGE64K::buf' is never used.	unusedStructMember	563

Brief description about static analysis :

- Variable Scope Issues: There are multiple warnings about the scope of variables, indicating that many variables (`ch`, `tmq`, `rio`, `stio`, etc.) have unnecessarily broad scopes. Reducing their scope could improve the maintainability and performance of the code.
- Shadowed Variables: One warning states that a local variable shadows another variable (`i`), which can lead to unexpected behavior or confusion.
- Const Pointer Declaration: Several warnings recommend declaring variables and parameters as pointers to `const`, which could enhance code safety by preventing unintended modifications to the data.
- Unused Struct Members: There are warnings about unused struct members (`TYPEUIDCWB:sqlCode`, `TYPEDBDIN:only`, etc.). These can either be removed to simplify the code or reviewed for relevance.
- General Style Issues: Most of the warnings are related to coding style, suggesting improvements for variable scope, pointer declarations, and unused elements to make the code more efficient, readable, and maintainable.