1. PROGRAM INSPECTION:

I have chosen this code from a github repository, whose link is given below:

https://github.com/godotengine/godot/blob/master/scene/animation/animation_node_state_machine.cpp

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

Category A: Data Reference Errors

Q . Does a referenced variable have a value that is unset or uninitialized?

A. Both AnimationNodeStateMachineTransition and AnimationNodeStateMachinePlayback classes have empty constructors.

Category E: Control-Flow Errors

Q. Will every loop eventually terminate? Devise an informal proof or argument showing that each loop will terminate.

A. The while (!found_route) loop relies on found_route being updated within the loop. If found_route is never set to true and open_list never becomes empty, it could result in an infinite loop

Q. For a loop controlled by both iteration and a Boolean condition (a searching loop, for example) what are the consequences of loop fall-through?

A. The code uses clear loop controls, but complex conditions like while (!found_route) require careful management. As previously mentioned, if found_route is not properly updated, loops could behave unpredictably.

Category H: Other Checks

Q. If the compiler produces a cross-reference listing of identifiers, examine it for variables that are never referenced or are referenced only once.

A. The paths list is populated with path_from and path_to but is **never used** within the method.

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

Solution:

All the categories test different things and hence for comprehensive testing/Inspection, all the catgories are essential. Hence, the best strategy is to **combine tests from multiple categories** to cover different types of errors.

The most common and small but important errors can be found using the inspection questions stated in categories A: **Data Reference** Errors, C: **Computation** Errors E: **Control-Flow** Errors, and F: **Interface Errors.**

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

Solution:

Errors such as segmentation fault , null pointer dereferencing, pointer point to random location and other such runtime errors cannot be found by this method. Also considering the code is big, it might also be the case that the person misses an error while going through the code.

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

Solution:

No, this program inspection technique is not worth applying as:

- For large codes, it takes a lot of time and is very tedious.
- It is to be done by a person and hence it is prone to human errors and understanding
- It also can't show logical errors in code for example if some formula is incorrect or not.

2. Debugging

Code 1 : Armstrong numbers

Errors 1.

Description - Illegal modifier for the method main; only public, protected, private, abstract, static, final, synchronized, native & strictfp are permitted

Breakpoints : 1

Solution: Added a package and module Armstrong in which Armstrong.java file is stored.

Error 2:

Description - Syntax error, insert "}" to complete ClassBody

Breakpoints : 1

Solution : Added terminating bracket

Error 3:

Description: Could not find or load main class Armstrong.Main in module ArmstrongModule

Breakpoints : 1

Solution: Added a main class with relevant main method

Complete Executable code:

```
package Armstrong;
import java.util.Scanner;
class Armstrong {
    public static void main(String args[]) {
        int num = Integer.parseInt(args[0]); // Input number
        int n = num; // Store original number for later comparison
        int check = 0, remainder;

        while(num > 0) {
            remainder = num % 10; // Correctly extract the last digit
            check = check + (int) Math.pow(remainder, 3); // Cube the remainder
and add to check
            num = num / 10; // Remove the last digit from num
        }

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

class Main{
        public static void main(String args[])
        {
               Scanner myObj = new Scanner(System.in);
                  System.out.println("Enter a number:\n");

                  String input[] = {"0"};
                  input[0] = myObj.nextLine();
                  Armstrong.main(input);


        }
}
```

Code 2 : GCD and LCM.java

Error 1:

Description : The public type GCD_LCM must be defined in its own file

Breakpoint: 1

Solution : Rename file to GCD_LCM.java

Error 2:

Description: java.lang.ArithmeticException: / by zero

Solution: while(a % b != 0) instead of while(a % b == 0)

Breakpoint: 1

Error 3: Logical error in lcm

Solution : if(a % x == 0 && a % y == 0) instead of if(a % x != 0 && a % y != 0)

Breakpoint: 1

Error 4: Logic error in gcd

Solution: a = (x < y) ? y : x; instead of a = (x > y) ? y : x;

Breakpoint: 1

Complete executable code:

```java
package SE_lab;
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; // a is greater number
        b = (x < y) ? x : y; // b is smaller number

        r = b;
        while(a % b != 0) //Error replace it with 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; // a is greater number
        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();
    }
}
```

Code 3 : Knapsack.java

Error 1:

Description: java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds.

Breakpoints : 1

Solution: Added command line arguments.

Error 2:

Description: java.lang.ArrayIndexOutOfBoundsException: Index -1787 out of bounds for length 2001

Breakpoints: 3

Solution:

if (weight[n] <= w) option2 = profit[n] + opt[n-1][w-weight[n]]; instead of if (weight[n] > w) option2 = profit[n-2] + opt[n-1][w-weight[n]];

Changed int option1 = opt[n-1][w]; from int option1 = opt[n++][w];

Complete executable code

```
//Knapsack
package SE_lab;
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-1][w];

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

                // 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]);
        }
    }
}
```

Code 4: Magic Number Check

Error 1:

Description : The public type MagicNumberCheck must be defined in its own file

Breakpoints required: 0

Solution : Rename file to MagicNumberCheck.java

Error 2:

Description : Syntax error, insert ";" to complete Statement

Breakpoints required: 0

Solution: Added semicolon there.

Warning 1:

Breakpoints required: 0

Description : Resource leak: 'ob' is never closed.

Solution: added ob.close()

Error 3:

Description: Incorrect logic

Breakpoints required: 1

Solution : while(sum>0) instead of while(sum==0)

Error 4:

Description: Incorrect logic

Breakpoints required: 1

Solution: s +=sum%10; instead of s = s*(sum/10);

Error 5:

Description: Incorrect Logic

Breakpoints required: 1

Solution: sum=(sum/10); instead of sum=(sum%10);

Complete Executable code:

```java
// Program to check if number is Magic number in JAVA
package SE_lab;
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();
        ob.close();
        int sum=0,num=n;
        while(num>9)
        {
            sum=num;
            int s=0;
            while(sum>0)
            {
                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.");
        }
    }
}
```

Code 5: MergeSort

Error 1:

Description : The public type MergeSort must be defined in its own file

Breakpoints required: 0

Solution : Rename file to MergeSort.java

Error 2:

Description: Type mismatch: cannot convert from int[] to int

Breakpoints required: 1

Solution:

```
int[] left = leftHalf(array);
int[] right = rightHalf(array);
```

instead of

```
int[] left = leftHalf(array+1);
int[] right = rightHalf(array-1);
```

Error 3:

Description: Type mismatch: cannot convert from int[] to int

Breakpoints required: 1

Solution: *merge*(array, left, right); instead of *merge*(array, left++, right--);

Complete Executable code:

```
// This program implements the merge sort algorithm for
// arrays of integers.
package SE_lab;

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);
            int[] right = rightHalf(array);

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

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

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



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

Code 6: Matrix Multiplication

Error 1:

Description : The public type MatrixMultiplication must be defined in its own file

Breakpoints required: 0

Solution : Rename file to MatrixMultiplication.java

Error 2:

Description: java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 2

Breakpoints required: 1

Solution: sum = sum + first[c][k]*second[k][d];

instead of

sum = sum + first[c-1][c-k]*second[k-1][k-d];

Complete executable code:

```
//Java program to multiply two matrices
package SE_lab;
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 < p ; k++ )
               {
                  sum = sum + first[c][k]*second[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");
         }
      }
   }
}
```

Code 7: Quadratic Probing

Error 1:

Description : The public type QuadraticProbingHashTableTest must be defined in its own file

Breakpoints required: 0

Solution : Rename file to QuadraticProbingHashTableTest.java

Error 2:
Description : Syntax error on tokens, they can be merged to form +=

Breakpoints : 0

Solution : + = replaced with =

Error 3:

Description: Logical Error

Breakpoints : 1

Solution: i = (i + h / h++) instead of i = (i + h / h--)

Error 4:

Description : Logical Error

Breakpoints : 1

Solution:

```
for (i = (i + h * h++) % maxSize; keys[i] != null; i = (i + (h = 1) * h++) %
maxSize)
```

instead of

```
for (i = (i + h * h++) % maxSize; keys[i] != null; i = (i + * h++) % maxSize)
```

Complete Executable Code:

```
    /**
    *   Java Program to implement Quadratic Probing Hash Table
     **/
    package SE_lab;
    import java.util.Scanner;
    /** Class QuadraticProbingHashTable **/
```

```
class QuadraticProbingHashTable
{
    private int currentSize, maxSize;
    private String[] keys;
    private String[] vals;

    /** Constructor **/

    public QuadraticProbingHashTable(int capacity)
    {
        currentSize = 0;
        maxSize = capacity;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    /** Function to clear hash table **/
    public void makeEmpty()
    {
        currentSize = 0;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    /** Function to get size of hash table **/
    public int getSize()
    {
        return currentSize;
    }

    /** Function to check if hash table is full **/
    public boolean isFull()
    {
        return currentSize == maxSize;
    }

    /** Function to check if hash table is empty **/
    public boolean isEmpty()
    {
        return getSize() == 0;
    }

    /** Fucntion to check if hash table contains a key **/
    public boolean contains(String key)
    {
        return get(key) !=  null;
    }
    /** Functiont to get hash code of a given key **/
    private int hash(String key)
    {
        return key.hashCode() % maxSize;
    }

    /** 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;
                    currentSize++;
                    return;
                }
                if (keys[i].equals(key))
                {
                    vals[i] = val;
                    return;
                }
                i = (i + h / h++) % maxSize;
            } while (i != tmp);
        }

        /** Function to get value for a given key **/

        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;
                System.out.println("i "+ i);
            }
            return null;
        }
        /** Function to remove key and its value **/

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

            /** find position key and delete **/
            int i = hash(key), h = 1;
            while (!key.equals(keys[i]))
                i = (i + h * h++) % maxSize;
            keys[i] = vals[i] = null;

            /** rehash all keys **/

            for (i = (i + h * h++) % maxSize; keys[i] != null; i = (i + (h = 1) *
h++) % maxSize)
            {
                String tmp1 = keys[i], tmp2 = vals[i];
                keys[i] = vals[i] = null;
                currentSize--;
                insert(tmp1, tmp2);
            }
            currentSize--;
        }


        /** Function to print HashTable **/
        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();
        }
    }

    /** Class QuadraticProbingHashTableTest **/
    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");
            /** maxSizeake object of QuadraticProbingHashTable **/
            QuadraticProbingHashTable qpht = new
QuadraticProbingHashTable(scan.nextInt() );

            char ch;
            /**  Perform QuadraticProbingHashTable operations  **/
            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;
                }

                /** Display hash table **/

                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');
        }
    }
```

Code 8: Sorting

Error 1:

Description : The public type Ascending_Order must be defined in its own file

Breakpoints required: 0

Solution : Rename file to Ascending_Order.java

Error 2:

Description : i cannot be resolved to a variable

Breakpoint : 1

Solution: Removed semicolon from this loop statement for (int i = 0; i >= n; i++);

Error 3:

Description: Logical Error

Breakpoints: 1

Solution: for (int i = 0; i < n; i++) instead of for (int i = 0; i >= n; i++)

Error 4:

Description : Logical Error

Breakpoints : 1

Solution : if (a[i] > a[j]) instead of if (a[i] <= a[j])

Complete Executable Code:

```
 // sorting the array in ascending order
package SE_lab;
import java.util.Scanner;
public class Ascending_Order
{
    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++)
        {
            for (int j = i + 1; j < n; j++)
            {
                if (a[i] > a[j])
                {
                    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]);
    }
}

// Input: Enter no. of elements you want in array: 5
//        Enter all elements:
//        1 12 2 9 7
//        1 2 7 9 12
```

Code 9 : StackMethods

Error 1:

Description : The public type StackReviseDemo must be defined in its own file

Breakpoints required: 0

Solution : Rename file to StackReviseDemo.java

Error 2:

Description: The public type StackMethods must be defined in its own file.

Breakpoints: 1

Solution: Made StackMethods a default class instead of public.

Error 2:

Description: Incorrect Logic. Decreasing top in push instead of increasing it.

Solution : top++ instead of top --

Error 3: Incorrect Logic

Description: Incorrect Logic.

Solution : for(int i=0;i<=top;i++) instead of for(int i=0;I > top;i++)

Error 4:

Description : Incorrect Logic. Increasing top in pop instead of Decreasing it.

Solution : top-- instead of top ++

Complete Executable Code:

```
//Stack implementation in java
package SE_lab;
import java.util.Arrays;

 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++;
            stack[top]=value;
        }
    }

    public void pop(){
        if(!isEmpty())
            top--;
        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++){
            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();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.pop();
        newStack.display();
    }
}
```

Code 10 : Tower of Hanoi

Error 1:

Description: The public type TowerOfHanoi must be defined in its own file

Breakpoints required: 0

Solution : Rename file to TowerOfHanoi.java

Error 2:

Description: Syntax error, insert ";" to complete Statement

Breakpoints : 0

Solution: Added semicolon.

Error 2:

Description : Syntax and Logical Error

Breakpoints : 2

Solution : *doTowers*(topN-1, inter, from, to); from doTowers(topN ++, inter--, from+1, to+1)

Complete Executable Code

```
//Tower of Hanoi
package SE_lab;
public class TowerOfHanoi{
   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);
```

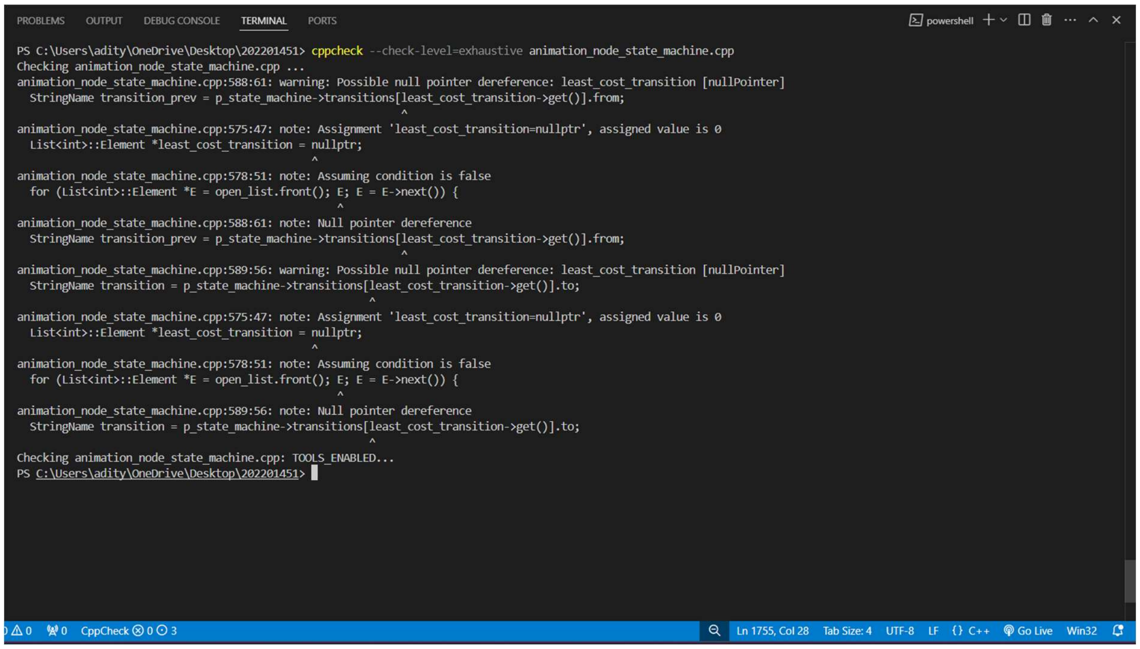
```
            doTowers(topN-1, inter, from, to);
        }
    }
}
```

3. Static analysis:

I have chosen this code from a github repository, whose link is given below:

https://github.com/godotengine/godot/blob/master/scene/animation/animation_node_state_machine.cpp

I have used CppCheck for the static analysis of the code. The output of the same is given in the Image below.



```
PS C:\Users\adity\OneDrive\Desktop\202201451> cppcheck --check-level=exhaustive animation_node_state_machine.cpp
Checking animation_node_state_machine.cpp ...
animation_node_state_machine.cpp:588:61: warning: Possible null pointer dereference: least_cost_transition [nullPointer]
  StringName transition_prev = p_state_machine->transitions[least_cost_transition->get()].from;
                                                            ^
animation_node_state_machine.cpp:575:47: note: Assignment 'least_cost_transition=nullptr', assigned value is 0
  List<int>::Element *least_cost_transition = nullptr;
                                              ^
animation_node_state_machine.cpp:578:51: note: Assuming condition is false
  for (List<int>::Element *E = open_list.front(); E; E = E->next()) {
                                                  ^
animation_node_state_machine.cpp:588:61: note: Null pointer dereference
  StringName transition_prev = p_state_machine->transitions[least_cost_transition->get()].from;
                                                            ^
animation_node_state_machine.cpp:589:56: warning: Possible null pointer dereference: least_cost_transition [nullPointer]
  StringName transition = p_state_machine->transitions[least_cost_transition->get()].to;
                                                       ^
animation_node_state_machine.cpp:575:47: note: Assignment 'least_cost_transition=nullptr', assigned value is 0
  List<int>::Element *least_cost_transition = nullptr;
                                              ^
animation_node_state_machine.cpp:578:51: note: Assuming condition is false
  for (List<int>::Element *E = open_list.front(); E; E = E->next()) {
                                                  ^
animation_node_state_machine.cpp:589:56: note: Null pointer dereference
  StringName transition = p_state_machine->transitions[least_cost_transition->get()].to;
                                                       ^
Checking animation_node_state_machine.cpp: TOOLS_ENABLED...
PS C:\Users\adity\OneDrive\Desktop\202201451>
```