

LAB-7

IT314

**USING GITHUB CODES & JAVA
PROGRAMS CODE INSPECTION,
DEBUGGING & STATIC ANALYSIS
TOOL**

Dharmi Patel

202201467

**Program Inspection/Debugging for Long-code from
GitHub**

We are given the following checklist and we have to find all the possible errors accordingly,

1. Data referencing Errors
2. Data declaration Errors
3. Computation Errors
4. Comparison Errors
5. Control Flow errors
6. Interface errors
7. Input/Output Errors
8. Other Checks

Category A: Data Reference Errors

- **File Handling:**

No file handling is observed in the current part of the code, but if file I/O is included elsewhere, ensure that proper error handling (e.g., `try-except` blocks) is used to catch potential exceptions like file access errors. Ensure all files are closed properly or use context managers (`with` statement) to manage file resources.

Category B: Data Declaration Errors

- **Data Initialization:**

Variables such as `weights`, `biasForVisibleUnits`, and `biasForHiddenUnits` are initialized properly using random number generators. However, ensure that any other variables introduced later in the code are initialized explicitly before use to avoid potential issues with uninitialized variables.

Category C: Computation Errors

- **Weight and Bias Initialization:**

The weights and biases are computed using a uniform distribution. Make sure that the boundaries for these distributions (based on the number of visible and hidden units) are set correctly to avoid incorrect initialization that might lead to training issues.

Category E: Control-Flow Errors

- **Excessive Use of Goto Statements:**

Python does not support `goto` statements, and this pattern is not present in the observed portion of the code. However, ensure that control flow is managed clearly with appropriate use of functions, loops, and conditionals instead of overly complex structures that could be replaced with simpler loops or helper functions.

Category G: Input/Output Errors

- **Inconsistent File Closing:**

There are no explicit file operations in the inspected part of the code. However, if there are sections where files are opened, ensure files are always closed, or use the `with` statement to handle file I/O safely to prevent resource leaks.

DEBUGGING:

1. Armstrong Number Program

- **Error:** Incorrect computation of the remainder.
- **Fix:** Use breakpoints to check the remainder calculation.

Corrected Code:

```
class Armstrong {  
    public static void main(String args[]) {  
        int num = Integer.parseInt(args[0]);  
        int n = num, check = 0, remainder;  
        while (num > 0) {  
            remainder = num % 10;  
            check += Math.pow(remainder, 3);  
            num /= 10;  
        }  
        if (check == n) {  
            System.out.println(n + " is an Armstrong Number");  
        } else {  
            System.out.println(n + " is not an Armstrong Number");  
        }  
    }  
}
```

2. GCDandLCMProgram

- **Errors:**

1. Incorrect while loop condition in GCD.
2. Incorrect LCM calculation logic.

- **Fix:**

Breakpoints at the GCD loop and LCM logic.

Corrected Code:

```
import java.util.Scanner;

public class GCD_LCM {
    static int gcd(int x, int y) {
        while (y != 0) {
            int temp = y;
            y = x%y;
            x =temp;
        }
        return x;
    }
    static int lcm(int x, int y) {
        return (x * y) / gcd(x, y);
    }
    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. Knapsack Program

- **Error:** Incrementing n inappropriately in the loop.
- **Fix:** Breakpoint to check loop behavior.

Corrected Code:

```

public class Knapsack {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int W=Integer.parseInt(args[1]);
        int[] profit = new int[N + 1], weight = new int[N + 1];
        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++) {
                int option1 = opt[n- 1][w];
                int option2 = (weight[n] <= w) ? profit[n] + opt[n- 1][w- weight[n]] :
                Integer.MIN_VALUE;
                opt[n][w] = Math.max(option1, option2);
                sol[n][w] = (option2 > option1);
            }
        }
    }
}

```

```
}  
}  
}  
}
```

4. Magic Number Program

- **Errors:**

1. Incorrect condition in the inner while loop.
2. Missing semicolons in expressions.

- **Fix:**

Set breakpoints at the inner while loop and check variable values.

Corrected Code:

```
import java.util.Scanner;  
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); // Fixed missing semicolon  
                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.");
    }
}
}

```

5. Merge Sort Program

- **Errors:**

1. Incorrect array splitting logic.
2. Incorrect inputs for the merge method.

- **Fix:** Breakpoints at array split and merge operations.

Corrected Code:

```

import java.util.Scanner;

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));
    }

    public static void mergeSort(int[] array) {

```



```
if (array.length > 1) {
    int[] le = le Half(array);
    int[] right = rightHalf(array);
    mergeSort(le );
    mergeSort(right);
    merge(array, le , right);
}
}

public static int[] le Half(int[] array) {
    int size1 = array.length / 2;
    int[] le = new int[size1];
    System.arraycopy(array, 0, le , 0, size1);
    return le ;
}

public static int[] rightHalf(int[] array) {
    int size1 = array.length / 2;
    int size2 = array.length- size1;
    int[] right = new int[size2];
    System.arraycopy(array, size1, right, 0, size2);
    return right;
}

public static void merge(int[] result, int[] le , int[] right) {
    int i1 = 0, i2 = 0;
    for (int i = 0; i < result.length; i++) {
```

```

    if (i2 >= right.length || (i1 < le .length && le [i1] <= right[i2])) {
        result[i] = le [i1];
        i1++;
    } else {
        result[i] = right[i2];
        i2++;
    }
}
}
}
}

```

6. Multiply Matrices Program

● Errors:

1. Incorrect loop indices.
2. Wrong error message.

● **Fix:** Set breakpoints to check matrix multiplication and correct messages.

Corrected 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 the first
    matrix");

```

```
m=in.nextInt();
n =in.nextInt();
int first[][] = new int[m][n];
System.out.println("Enter the elements of the first matrix");
for (c = 0; c < m; c++)
for (d = 0; d < n; d++)
f
irst[c][d] = in.nextInt();
System.out.println("Enter the number of rows and columns of the
second matrix");
p =in.nextInt();
q =in.nextInt();
if (n != p)
System.out.println("Matrices with entered orders can't be
multiplied.");
else {
int second[][] = new int[p][q];
int multiply[][] = new int[m][q];
System.out.println("Enter the elements of the 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+=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");
}
}
}

```

7. Quadratic Probing Hash Table Program

- Errors:

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

- Fix: Set breakpoints and step through logic for insert, remove, and get methods.

Corrected Code:

```
import java.util.Scanner;
```

```
class QuadraticProbingHashTable {
    private int currentSize, maxSize;
    private String[] keys, vals;
    public QuadraticProbingHashTable(int capacity) {
        currentSize = 0;
        maxSize = capacity;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }
    public void insert(String key, String val) {
        int tmp = hash(key), 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 += (h * h++) % maxSize;
        } 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;  
    }  
    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;  
}  
  
private boolean contains(String key) {  
    return get(key) != null;  
}  
  
private int hash(String key) {  
    return key.hashCode() % maxSize;  
}  
}
```

```
public class HashTableTest {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        QuadraticProbingHashTable hashTable = new  
        QuadraticProbingHashTable(scan.nextInt());  
        hashTable.insert("key1", "value1");  
        System.out.println("Value: " + hashTable.get("key1"));  
    }  
}
```

8. Sorting Array Program

- Errors:

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

- Fix: Set breakpoints to check the loop and class name.

Corrected Code:

```
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 the number of elements: ");  
        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.println("Sorted Array: " + Arrays.toString(a));
}
}

```

9. Stack Implementation Program

- Errors:

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

- Fix: Add breakpoints to check push, pop, and display methods.

Corrected Code:

```

public class StackMethods {
    private int top;
    private int[] stack;
    public StackMethods(int size) {

```



```
stack = new int[size];
top = -1;
}
public void push(int value) {
    if (top == stack.length - 1) {
        System.out.println("Stack full");
    } else {
        stack[++top] = value;
    }
}
public void pop() {
    if (top == -1) {
        System.out.println("Stack empty");
    } else {
        top--;
    }
}
public void display() {
    for (int i = 0; i <= top; i++) {
        System.out.print(stack[i] + " ");
    }
    System.out.println();
}
}
```

10. Tower of Hanoi Program

- Error: Incorrect increment/decrement in recursive call.
- Fix: Breakpoints at the recursive calls to verify logic.

Corrected Code:

```
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);  
        }  
    }  
}
```

```
"c:\Users\dharm\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher\..\..\debugpy\__main___.py", line 39, in <module>
    cli.main()
```

File

"c:\Users\dharm\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher\..\..\debugpy\..\debugpy\server\cli.py", line 430, in main

run()

File

"c:\Users\dharm\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher\..\..\debugpy\..\debugpy\server\cli.py", line 284, in run_file

```
runpy.run_path(target, run_name="__main__")
```

File

```
"c:\Users\dharm\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 320, in run_path
```

```
code, fname = get code from file(run name, path name)
```


File

"c:\Users\dharm\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy_vendored\pydevd_pydevd_bundle\pydevd_runpy.py", line 294, in get_code_from_file

```
code = compile(f.read(), fname, 'exec')
```

ΛΛΛ

File "C:\Users\dharm\Desktop\SEM

5\SE\Testing lab\RestrictedBoltzmannMachines.py", line 69

print 'PARAMETER SIZE MISMATCH INSIDE'

^^

SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?

