IT 314 - Software Engineering

Lab - 7

Name : Ramani Priyank
ID : 202201497

II. CODE DEBUGGING: Debugging is the process of localizing, analyzing, and removing suspected errors in the code.

Code – 1: Armstrong Number

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

Remainder calculation: `remainder = num / 10` should be `remainder = num % 10`.
Number reduction: `num = num % 10` should be `num = num / 10`.

2. How many breakpoints you need to fix those errors?
   a. What are the steps you have taken to fix the error you identified in the code fragment?

Two breakpoints:

1. Change `remainder = num / 10` to `remainder = num % 10`.
2. Change `num = num % 10` to `num = num / 10`.

3. 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 += (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");
    }
}
```

Code – 2 : GCD and LCM

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

Error in the GCD loop: The condition `while(a % b == 0)` is incorrect. It should be `while(a % b != 0)`, as the loop needs to continue until `a % b` equals 0.
Error in the LCM calculation: The condition `if(a % x != 0 && a % y != 0)` is incorrect. It should be `if(a % x == 0 && a % y == 0)` to check when both `a` is divisible by `x` and `y`.

2. How many breakpoints you need to fix those errors?
   a. What are the steps you have taken to fix the error you identified in the code fragment?

Two breakpoints:

1. Change `while(a % b == 0)` to `while(a % b != 0)`.
2. Change `if(a % x != 0 && a % y != 0)` to `if(a % x == 0 && a % y == 0)`.

3. Submit your complete executable code?

```java
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 the smaller number
        b = (x < y) ? x : y; // b is the larger number

        r = b;
        while(a % b != 0) // Corrected the condition
        {
            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 the greater number
    while(true)
    {
        if(a % x == 0 && a % y == 0) // Corrected the condition
            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

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

Incrementing `n` in the option1 calculation: In the line `int option1 = opt[n++][w];`, the `n++` should not be there. It increments `n`, which causes the program to skip some iterations. It should be `opt[n][w]`.
Wrong index in option2 calculation: In the line `if (weight[n] > w) option2 = profit[n-2] + opt[n-1][w-weight[n]];`, `profit[n-2]` should be `profit[n]`. This error refers to the wrong index for profit calculation when selecting the item.

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

Two breakpoints:

1. Change `int option1 = opt[n++][w];` to `int option1 = opt[n][w];`.
2. Change `profit[n-2]` to `profit[n]` in `option2` calculation.

3. Submit your complete executable 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];
        int[] weight = new int[N+1];

        for (int n = 1; n <= N; n++) {
            profit[n] = (int) (Math.random() * 1000);
            weight[n] = (int) (Math.random() * W);
        }

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

                opt[n][w] = Math.max(option1, option2);
                sol[n][w] = (option2 > option1);
            }
        }
```

```
        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;
            }
        }

        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

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

Logical issue in inner while loop: The condition `while(sum == 0)` should be `while(sum > 0)`. This ensures the loop continues while `sum` is greater than 0.
Multiplication instead of addition: The line `s=s*(sum/10)` should be `s=s + (sum % 10)` to correctly sum up the digits.
Missing semicolon: There's a missing semicolon after `sum=sum%10`.

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

Three breakpoints:

1. Change `while(sum == 0)` to `while(sum > 0)`.
2. Change `s=s*(sum/10)` to `s=s + (sum % 10)`.
3. Add a semicolon after `sum=sum%10`.

3. Submit your complete executable code?

```java
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 condition
            {
                s = s + (sum % 10); // Corrected operation
                sum = sum / 10; // Added 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.");
        }
    }
```

```java
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 += (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");
    }
}
```

Code – 5 : Merge sort

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

Array slicing in `mergeSort`: The expressions `leftHalf(array+1)` and `rightHalf(array-1)` are incorrect because they are attempting to perform arithmetic on arrays. They should pass the entire array as an argument instead.
Incorrect increment/decrement in merge function: The expressions `merge(array, left++, right--)` should not increment or decrement. Instead, pass `left` and `right` directly without modifying them.

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

Two breakpoints:

1. Change `leftHalf(array+1)` to `leftHalf(array)` and `rightHalf(array-1)` to `rightHalf(array)`.

2. Change `merge(array, left++, right--)` to `merge(array, left, right)`.

3. Submit your complete executable code?

```java
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));
    }

    public static void mergeSort(int[] array) {
        if (array.length > 1) {
            int[] left = leftHalf(array); // Corrected argument
            int[] right = rightHalf(array); // Corrected argument

            mergeSort(left);
            mergeSort(right);

            merge(array, left, right); // Removed unnecessary increments
        }
    }

    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;
    }
```

```
    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;
    }

    public static void merge(int[] result, int[] left, int[] right) {
        int i1 = 0;
        int i2 = 0;

        for (int i = 0; i < result.length; i++) {
            if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) {
                result[i] = left[i1];
                i1++;
            } else {
                result[i] = right[i2];
                i2++;
            }
        }
    }
}
```

Code – 6 : Multiply matrix

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

Incorrect indexing in matrix multiplication loop: The expressions `first[c-1][c-k]` and `second[k-1][k-d]` are incorrect. Matrix indices should not be decremented.
Incorrect loop limit for `k`: The variable `k` should loop from `0` to `n` (the number of columns in the first matrix), not `p` (number of rows of the second matrix).

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

Two breakpoints:

1. Change `first[c-1][c-k]` to `first[c][k]`.
2. Change `second[k-1][k-d]` to `second[k][d]` and modify the loop limit for `k` to `n`.

3. Submit your complete executable code?

```
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
        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++) {
                        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 : QuadraticProbing

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

Error 1: The line `i + = (i + h / h--) % maxSize;` contains a syntax error with an extra space. It should be `i += (tmp + h * h) % maxSize;`.
Error 2: In the `get` method, the line `i = (i + h * h++) % maxSize;` should increment `h` correctly before using it for probing.

Error 3: In the `remove` method, the rehashing logic has an incorrect incrementing of `h`, which could lead to an infinite loop.
Error 4: The `makeEmpty` method resets `keys` and `vals` without releasing memory, which could lead to memory issues (though not strictly an error).

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

Total Breakpoints Needed: 3

- Breakpoint 1: In the `insert` method to check how i is updated and ensure it properly probes for a new position.
- Breakpoint 2: In the `get` method to monitor the value of `i` and confirm proper retrieval.
- Breakpoint 3: In the `remove` method to ensure the correct deletion of keys and rehashing.

3. Submit your complete executable code?

```
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;
```

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

private int hash(String key) {
    return key.hashCode() % 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;
        h++;
    } while (i != tmp);
}
```

```java
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;
        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");
```

```
            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');
    }
```

Code – 8 : Sorting Array

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

Class name `Ascending _Order` contains an invalid space.
The loop condition in `for (int i = 0; i >= n; i++)` should be `i < n`.
The semicolon after the first `for` loop causes incorrect behavior.

The sorting condition `if (a[i] <= a[j])` should be `if (a[i] > a[j])` for ascending order.

2. How many breakpoints you need to fix those errors?
    a. What are the steps you have taken to fix the error you identified in the code fragment?

4 breakpoints required:

1. At the class definition `public class Ascending _Order`.
2. At the loop condition `for (int i = 0; i >= n; i++)`.
3. At the semicolon after the first `for` loop.
4. At the swap condition `if (a[i] <= a[j])`.

3. Submit your complete executable code?

```java
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++) {
            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]);
    }
}
```

Code – 9 : Stack Implementation

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

In the `push` method, `top--` should be `top++` to push values correctly.
In the `pop` method, `top++` should be `top--` to pop values correctly.
In the `display` method, the loop condition should be `i <= top` instead of `i > top` to display all elements.

2. How many breakpoints you need to fix those errors?
   a. What are the steps you have taken to fix the error you identified in the code fragment?

3 breakpoints required:

1. At the `push` method to check value assignment.
2. At the `pop` method to check top index decrement.
3. At the `display` method loop to check the loop condition.

3. Submit your complete executable code?

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

    public void pop() {
        if (!isEmpty()) {
            int poppedValue = stack[top];
            top--;
            System.out.println("Popped value: " + poppedValue);
        } else {
            System.out.println("Can't pop...stack is empty");
        }
    }
```

```
    public boolean isEmpty() {
        return top == -1;
    }

    public void display() {
        if (isEmpty()) {
            System.out.println("Stack is empty.");
            return;
        }
        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

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

In the `doTowers` method, the recursive call `doTowers(topN ++, inter--, from+1, to+1)` contains incorrect increment and decrement operations. It should be `topN - 1`, `inter`, `from`, and `to`.
The characters `from+1` and `to+1` should not be incremented as they are intended to be characters.
The base case does not handle the printing of the disk move correctly when there is only one disk.

2. How many breakpoints you need to fix those errors?
   a. What are the steps you have taken to fix the error you identified in the code fragment?

3 breakpoints required:

1. At the base case in `doTowers` to check if it is reached correctly.
2. At the first recursive call to ensure the correct parameters are passed.
3. At the second recursive call to verify the parameters and ensure they are not incorrectly modified.
3.

3. 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);
        }
    }
}
```

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.

- **Category A**: Data Reference Errors
  - Uninitialized Variables: The variable `r` in the `gcd` method is initialized to 0 but may not be set correctly if the loop does not execute.
  - Incorrect Variable Usage: The variables `a` and `b` are used without proper initialization in some cases, leading to potential undefined behavior.
- **Category B**: Computation Errors
  - Incorrect Loop Condition: In the `gcd` method, the while loop condition should be `while (a % b != 0)` instead of `while (a % b == 0)`, which would cause an infinite loop if `b` is greater than zero.
  - Potential Infinite Loop: In the `lcm` method, if both inputs are zero, the loop will run indefinitely as it checks for divisibility by zero.
- **Category C:** Input/Output Errors
  - No Input Validation: The program does not validate user input, which could lead to runtime errors if non-integer values are provided.
- **Category F:** Interface Errors
  - Incorrect Parameter Handling: The method definitions assume specific input formats without validating them, which could lead to unexpected behavior.
- **Category G:** Input/Output Errors
  - Missing Error Handling for Input Data: The program does not handle non-square matrices or incompatible dimensions for operations.

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

- **Category A**: Data Reference Errors would be the most effective for inspection in this code. Ensuring that variables are properly initialized and utilized prevents many runtime errors and enhances code reliability.

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

- Logic Errors: Program inspection may not effectively identify logic errors where the code runs without crashing but produces incorrect results due to flawed algorithmic logic, such as incorrect GCD or LCM calculations under specific conditions.

4. Is the program inspection technique is worth applicable?

- Yes, program inspection is a valuable technique. It helps identify significant errors related to data references and computations early in the development process, improving code quality and maintainability. However, it should be complemented with testing methods like unit tests to ensure comprehensive coverage of potential errors and logical correctness.