C-DAC Mumbai

Date 25/09/2024

Subject: Algorithm and Data Structure
Assignment 1

Solve the assignment with following thing to be added in each question.

-Program
-Flow chart
-Explanation
-Output
-Time and Space complexity

1. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

Input: 153
Output: true
Input: 123
Output: false

D:\@CDAC\ADS\Assignment1\ArmstrongNum.java - Notepad++

```java
import java.util.*;

public class ArmstrongNum{

    public static boolean isArmstrong(int num){
        int originalnum = num;
        int result = 0;
        int pw = String.valueOf(originalnum).length();
        while(num!=0){
            int digit = num % 10;
            result += Math.pow(digit,pw);
            num /= 10;
        }
        if(result == originalnum){
            return true;
        }
        else{
            return false;
        }

    }
    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number : ");
        int number = sc.nextInt();
        System.out.println(isArmstrong(number));
    }
}
```

Java source file | length : 596 lines : 29 | Ln : 16 Col : 10 Pos : 345 | Windows (CR LF) UTF-8 INS

FlowChart:

```
Start
  |
  V
Input the number (n)
  |
  V
Set originalNumber = n
Set result = 0
Set numberOfDigits = number of digits in n
  |
  V
WHILE n != 0
  |     |
  |     V
  |  Extract the last digit (digit = n % 10)
  |     |
  |  Add digit^numberOfDigits to result (result += digit^numberOfDigits)
  |     |
  |  Remove last digit from n (n = n / 10)
  |
  V
IF result == originalNumber
  |     |
  |   return true
  |     |
  |  return false
  |
  V
End
```

Explanation:

Step 1: **Input**: The user enters a number n.
Step 2: **Initialize**: Store the original number for comparison, set result to 0, and calculate the number of digits in n.
Step 3: **Loop**:

- While there are still digits in n, extract the last digit.
- Add the digit raised to the power of the number of digits to the result.
- Remove the last digit from n.

Step 4: **Comparison**: After the loop, compare result to the original number.

- If they are equal, it is an Armstrong number.
- If not, it is not an Armstrong number.

Step 5: **End** the program.

Output:

```
C:\WINDOWS\system32\cmd.

Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

D:\@CDAC\ADS\Assignment1>javac ArmstrongNum.java

D:\@CDAC\ADS\Assignment1>java ArmstrongNum
Enter a number :
153
true

D:\@CDAC\ADS\Assignment1>java ArmstrongNum
Enter a number :
121
false
```

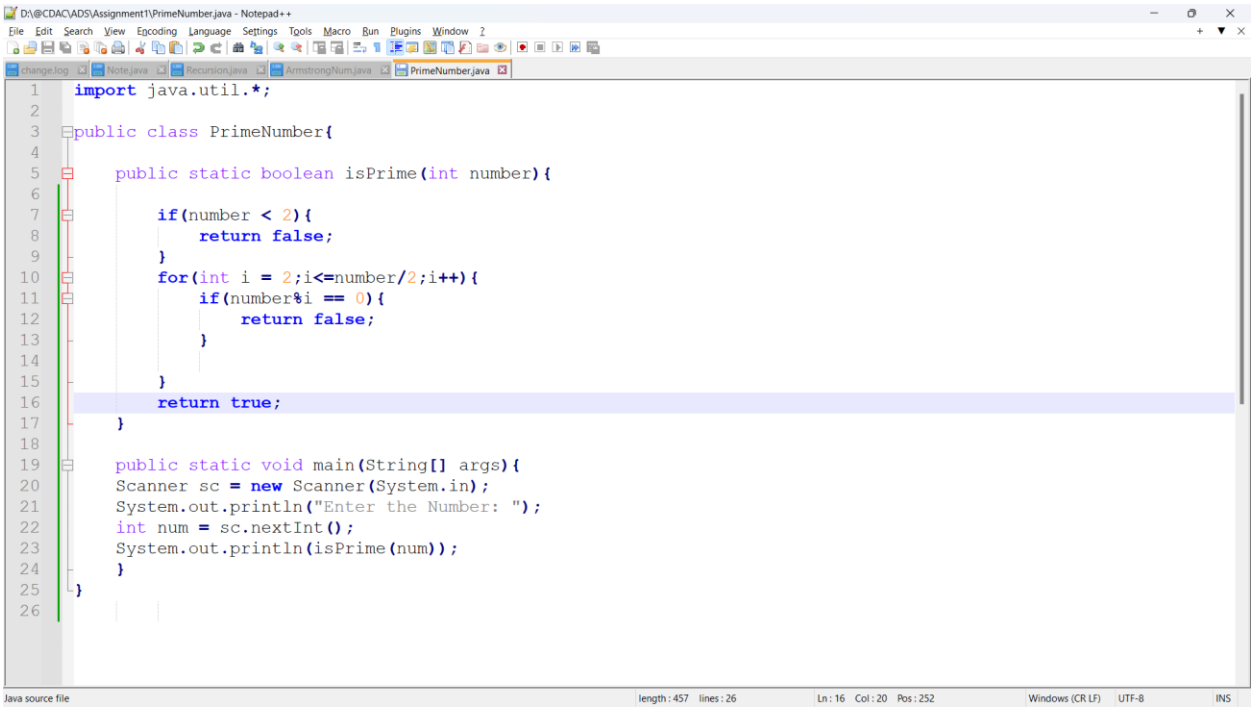
Time and Space Complexity is: O(logn).

2. Prime Number
Problem: Write a Java program to check if a given number is prime.

Test Cases:

Input: 29
Output: true
Input: 15
Output: false

Program



FlowChart:

```
Start
   |
   v
[Input Number]
   |
   v
[Is number < 2?]
   |
  Yes ----------------------------------> No
   |                                      |
[Print: "Not Prime"]                  [Loop: i = 2 to number / 2]
   |                                      |
  End                                 [Is number % i == 0?]
                                         |
                                     Yes |                          No
                                      |   v
                              [Return false]  ------------------
                                      |                              |
                                     End                          Loop Ends
                                                                   |
                                                                [Return True]
                                                                     |
                                                                    End
```

Explanation:

1. Input Number: The program prompts the user to input a number.
2. Is number < 2?: If the number is less than 2, the program outputs "Not Prime" and terminates, as numbers less than 2 are not prime.
3. Loop from 2 to number/2: The program iterates from i = 2 to number/2.
4. Is number divisible by i? (number % i == 0): For each i, the program checks if the number is divisible by i.

- **If it is divisible, it prints "Not Prime" and exits.**
- **If no divisors are found, the loop ends.**

5. End of Loop: If the loop completes without finding any divisors, the program prints "Prime" and terminates.

Output:

```
D:\@CDAC\ADS\Assignment1>javac primeNumber.java

D:\@CDAC\ADS\Assignment1>java primeNumber.java
Enter the Number:
29
true

D:\@CDAC\ADS\Assignment1>java primeNumber.java
Enter the Number:
15
false

D:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(n)
Space Complexity: O(1)

3. Factorial
Problem: Write a Java program to compute the factorial of a given number.

Test Cases:

Input: 5
Output: 120
Input: 0
Output: 1

D:\@CDAC\ADS\Assignment1\Factorial.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

change.log | Note.java | Recursion.java | ArmstrongNum.java | PrimeNumber.java | Factorial.java

```java
import java.util.*;

public class Factorial{

    public static int factorial(int num){
        int fact = 1;
        if(num <= 1){
            return 1;
        }

        for(int i = 2; i<= num; i++){
            fact *= i;
        }
        return fact;

    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number");
        int num = sc.nextInt();
        System.out.println(factorial(num));
    }
}
```

Java source file | length : 424 lines : 23 | Ln : 12 Col : 13 Pos : 186 | Windows (CR LF) | UTF-8 | INS

```
                +-------------------------------+
                |          Start                |
                +-------------------------------+
                                |
                                v
                +-------------------------------+
                | Prompt user to enter a number |
                +-------------------------------+
                                |
                                v
        +-------------------------------------------------------+
        | Check if the input is a valid integer (hasNextInt)|
        +-------------------------------------------------------+
             |                  |
             | (No)             | (Yes)
             v                  v
   +-----------------+    +-------------------------------+
   | Display "Invalid|    | Read the input number (num)   |
   | input message"  |    +-------------------------------+
   +-----------------+                   |
                                          v
                              +-------------------------+
                              | Is num < 0 ?            |
                              +-------------------------+
                             /                 \
                        (Yes)                  (No)
                           /                     \
           +---------------------+      +-------------------------------+
           | Print error message |      | Is num == 0 or num == 1 ?     |
           | "Factorial not      |      +-------------------------------+
           | defined for         |                  /       \
           | negative numbers"   |             (Yes)        (No)
           +---------------------+                 /         \
                                                  v           v
                          +---------------------+  +-----------------------------+
                          | Return 1 (Base case)|  | Initialize fact = 1         |
                          +---------------------+  +-----------------------------+
                                                   | Loop from 2 to num          |
                                                   +-----------------------------+
                                                              |
                                                              v
                                                   +-----------------------------+
                                                   | fact = fact * i (multiply)  |
                                                   +-----------------------------+
                                                              |
                                                              v
                                                   +-----------------------------+
                                                   | End loop and return fact    |
                                                   +-----------------------------+
                                                              |
                                                              v
                                          +-----------------------------------+
                                          | Display the calculated factorial |
                                          +-----------------------------------+
                                                              |
                                                              v
                                                   +---------------------+
                                                   |        End          |
                                                   +---------------------+
```

Explanation:

1. **Start**: The program begins.
2. **Input Handling**: The user is prompted to input a number.
3. **Input Validation**: The program checks whether the input is a valid integer:
   - If not valid, it prints an error message and ends.
   - If valid, the program proceeds.
4. **Negative Input Check**: It checks if the input number is negative:
   - If negative, an error message is displayed, and the program ends.
   - If not negative, it proceeds.
5. **Base Case Check**: The program checks if the number is 0 or 1 (since the factorial of these is 1):
   - If true, it returns 1.
6. **Factorial Calculation**: If the number is greater than 1:
   - The program initializes a variable fact to 1.
   - It loops through all numbers from 2 to the input number, multiplying them to fact.
7. **Display the Result**: The program prints the calculated factorial.
8. **End**: The program finishes.

Output:

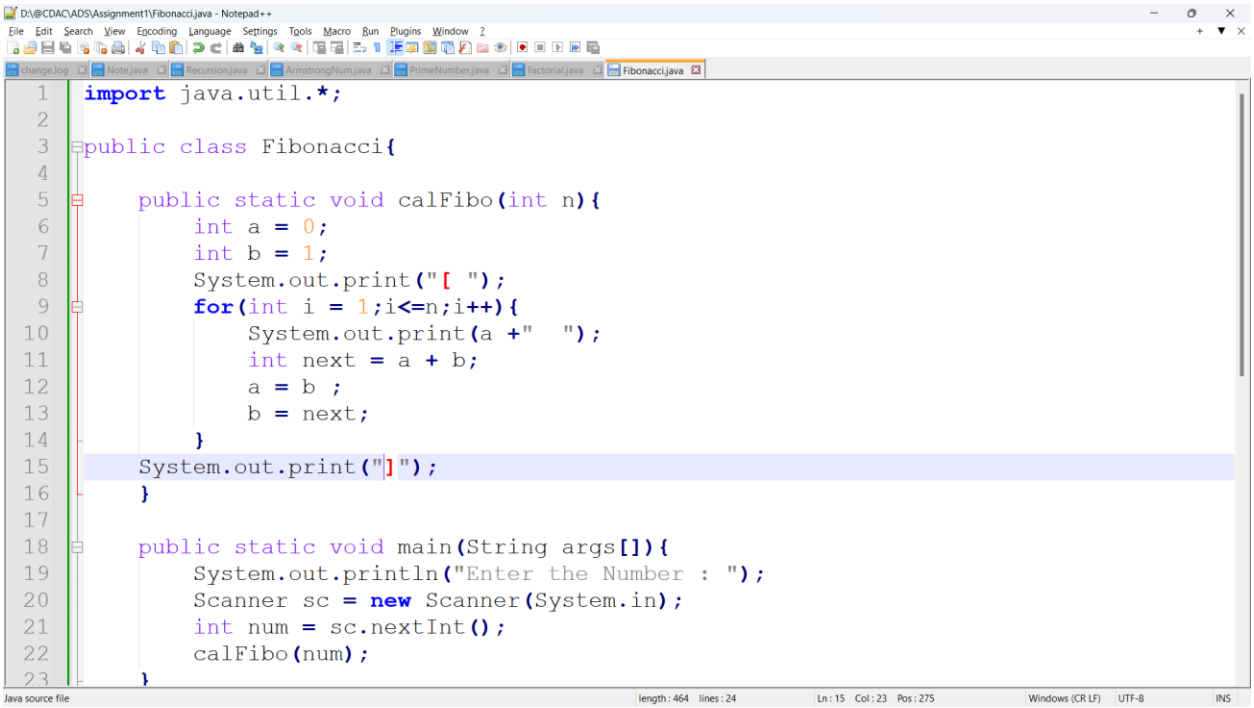
```
D:\@CDAC\ADS\Assignment1>java factorial
Error: Could not find or load main class factorial
Caused by: java.lang.NoClassDefFoundError: factorial (wrong

D:\@CDAC\ADS\Assignment1>java factorial.java
Enter the number
5
120

D:\@CDAC\ADS\Assignment1>java factorial.java
Enter the number
0
1

D:\@CDAC\ADS\Assignment1>
```

- **Time Complexity: O(num)**
- **Space Complexity: O(1)**

4. Fibonacci Series
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

Test Cases:

Input: n = 5
Output: [0, 1, 1, 2, 3]
Input: n = 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]



```
            +-------------------------------+
            |             Start             |
            +-------------------------------+
                           |
                           v
            +-------------------------------+
            | Prompt user to enter a number |
            +-------------------------------+
                           |
                           v
      +-------------------------------------------------+
      |             Read the input number (n)           |
      +-------------------------------------------------+
                           |
                           v
            +-------------------------------+
            |  Initialize a = 0, b = 1      |
            +-------------------------------+
                           |
                           v
      +-------------------------------------------------+
      |        Print the opening bracket "[ "           |
      +-------------------------------------------------+
                           |
                           v
            +-----------------------------------+
            |  Loop from i = 1 to n             |
            +-----------------------------------+
                           |
                           v
      +-------------------------------------------------+
      |       Print current value of 'a'                |
      +-------------------------------------------------+
                           |
                           v
      +-------------------------------------------------+
      |  Calculate next = a + b                         |
      |  Update a = b, b = next                         |
      +-------------------------------------------------+
                           |
                           v
            +-----------------------------------+
            |   End loop when i > n             |
            +-----------------------------------+
                           |
                           v
      +-------------------------------------------------+
      |        Print the closing bracket "]"            |
      +-------------------------------------------------+
                           |
                           v
            +-------------------------------+
            |             End               |
            +-------------------------------+
|
```

Explanation:

1. Start: The program starts.
2. Input Handling: The user is prompted to input a number n, and the input is read.
3. Initialization: The variables a (initial value 0) and b (initial value 1) are initialized.
4. Opening Bracket: The program prints the opening bracket [to indicate the start of the Fibonacci sequence.
5. Loop (i = 1 to n): The program enters a loop that runs n times:
   - Print current value of a: The current Fibonacci number (stored in a) is printed.
   - Next Fibonacci number: The next Fibonacci number is calculated as the sum of a and b.
   - Update: The values of a and b are updated: a = b, and b = next.
6. Loop End: The loop continues until i > n.
7. Closing Bracket: The program prints the closing bracket] to indicate the end of the Fibonacci sequence.
8. End: The program ends.

Output:

```
D:\@CDAC\ADS\Assignment1>javac fibonacci.java

D:\@CDAC\ADS\Assignment1>java fibonacci.java
Enter the Number :
5
[ 0  1  1  2  3  ]
D:\@CDAC\ADS\Assignment1>java fibonacci.java
Enter the Number :
8
[ 0  1  1  2  3  5  8  13  ]
D:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(n)
Space Complexity: O(1)

5. Find GCD
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

Test Cases:

Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

```java
import java.util.Scanner;

public class GCD {


    public static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.println("Enter the first number:");
        int a = sc.nextInt();
        System.out.println("Enter the second number:");
        int b = sc.nextInt();


        int gcd = findGCD(a, b);


        System.out.println("The GCD of " + a + " and " + b + " is: " + gcd);
    }
}
```

```
                +-------------------------+
                |         Start           |
                +-------------------------+
                           |
                           V
                +-------------------------+
                |  Input two numbers, a, b |
                +-------------------------+
                           |
                           V
                +-------------------------+
                |      While b ≠ 0        |
                +-------------------------+
                   /            \
           +-------Yes------+     No ------+
           |                |               |
           V                |               |
  +----------------+        |               |
  | temp = b       |        |               |
  +----------------+        |               |
  | b = a % b      |        |               |
  +----------------+        |               |
  | a = temp       |        |               |
  +----------------+        |               |
           |                |               |
           +----------------+---------------+
                           |
                           V
                +-------------------------+
                |  Print a (GCD of a and b)|
                +-------------------------+
                           |
                           V
                +-------------------------+
                |          End            |
                +-------------------------+
```

1. **Start**:
   - The program begins execution.
2. **Input Handling**:
   - The program prompts the user to input two numbers (a and b).
   - The user enters the two numbers.
3. **GCD Calculation (Using a while loop)**:
   - The program enters a while loop which continues as long as b is not equal to 0.
   - In each iteration of the loop:
     1. **Store current value of b** in a temporary variable (temp).
     2. **Calculate a % b** and assign the result to b.
     3. **Assign the old value of b** (stored in temp) to a.
4. **Loop Termination**:
   - When b becomes 0, the loop terminates.
   - The current value of a is the **GCD**.
5. **Output**:
   - The program prints the GCD of the two numbers (a).
6. **End**:
   - The program finishes execution.

Example Execution Flow:

Input: a = 54, b = 24

- **Initial values**: a = 54, b = 24
  - temp = 24
  - b = 54 % 24 = 6
  - a = 24
- **Next iteration**:

    - temp = 6
    - b = 24 % 6 = 0
    - a = 6
- **Loop ends** as b is now 0.
- **GCD = 6** is printed.

Output:

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

D:\@CDAC\ADS\Assignment1>javac gcd.java

D:\@CDAC\ADS\Assignment1>java gcd
Error: Could not find or load main class gcd
Caused by: java.lang.NoClassDefFoundError: gcd (wrong name: GCD)

D:\@CDAC\ADS\Assignment1>java gcd.java
Enter the first number:
54
Enter the second number:
24
The GCD of 54 and 24 is: 6

D:\@CDAC\ADS\Assignment1>java gcd.java
Enter the first number:
17
Enter the second number:
13
The GCD of 17 and 13 is: 1

D:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(log(min(a, b)))
Space Complexity: O(1)

6. Find Square Root
Problem: Write a Java program to find the square root of a given number (using integer approximation).

Test Cases:

Input: x = 16
Output: 4
Input: x = 27
Output: 5

D:\@CDAC\ADS\Assignment1\SquareRoot.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

change.log | Note.java | Recursion.java | ArmstrongNum.java | PrimeNumber.java | Factorial.java | Fibonacci.java | SquareRoot.java

```java
import java.util.Scanner;

public class SquareRoot {

    public static int squareRoot(int x) {
        if (x == 0 || x == 1) {
            return x;
        }

        int start = 1, end = x, result = 0;

        while (start <= end) {
            int mid = (start + end) / 2;


            if (mid * mid == x) {
                return mid;
            }


            if (mid * mid < x) {
                start = mid + 1;
                result = mid;
            }

            else {
                end = mid - 1;
            }
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = sc.nextInt();

        int squareRoot = squareRoot(number);

        System.out.println( squareRoot);
    }
}
```

Java source file | length : 910 lines : 44 | Ln : 41 Col : 28 Pos : 886 | Windows (CR LF) UTF-8 INS

```
+-------------------------------------+
|               Start                 |
+-------------------------------------+
                |
                v
+-------------------------------------+
|        Input x from the user        |
+-------------------------------------+
                |
                v
+-------------------------------------+
|   Check if x is 0 or 1 (Base Case)  |
+-------------------------------------+
       /                  \
     /Yes                 No\
    v                      v
+-----------------+   +-------------------------------------+
|  Return x (0/1) |   |    Initialize: start = 1, end = x   |
+-----------------+   |          result = 0                 |
                      +-------------------------------------+
                                     |
                                     v
                      +-------------------------------------+
                      |       While start <= end:           |
                      +-------------------------------------+
                                     |
                                     v
                      +-------------------------------------+
                      |     Calculate mid = (start + end) / 2 |
                      +-------------------------------------+
                                     |
                                     v
          +-------------------------------------+-----------------------------+
          |                                     |                             |
+-----------------------+   +-----------------------------+   +-----------------------------+
|  Check if mid * mid == x  |   |  Check if mid * mid < x  |   |   Check if mid * mid > x    |
+-----------------------+   +-----------------------------+   +-----------------------------+
          |                                     |                             |
      Yes (Exact root)                        Yes                           Yes
          v                                     v                             v
+-----------------------+   +-----------------------------+   +-----------------------------+
|   Return mid          |   | Set start = mid + 1         |   | Set end = mid - 1           |
+-----------------------+   | Store result = mid          |   +-----------------------------+
                            +-----------------------------+
                                     |
                                     v
                      +-------------------------------------+
                      |       End While Loop                |
                      +-------------------------------------+
                                     |
                                     v
                      +-------------------------------------+
                      |     Return result (Approximation)   |
                      +-------------------------------------+
                                     |
                                     v
                      +-------------------------------------+
                      |                End                  |
                      +-------------------------------------+
```

Explanation:

1. Start
    - Begin the program.
2. Input: x
    - Get the input number x from the user.
3. Check if x is 0 or 1
    - If x == 0 or x == 1, return x since the square root of 0 is 0 and the square root of 1 is 1.
4. Initialize start, end, and result
    - Set start = 1, end = x, and result = 0.
5. While start <= end
    - Continue while start is less than or equal to end.
6. Calculate mid
    - Calculate mid = (start + end) / 2.
7. Check if mid * mid equals x
    - If mid * mid == x, return mid.
8. Check if mid * mid is less than x
    - If mid * mid < x, set start = mid + 1 and store result = mid.
9. Else (mid * mid > x)
    - If mid * mid > x, set end = mid - 1.
10. End loop
11. Return result

- Return the value of result as the integer approximation of the square root.

12. End

Output:

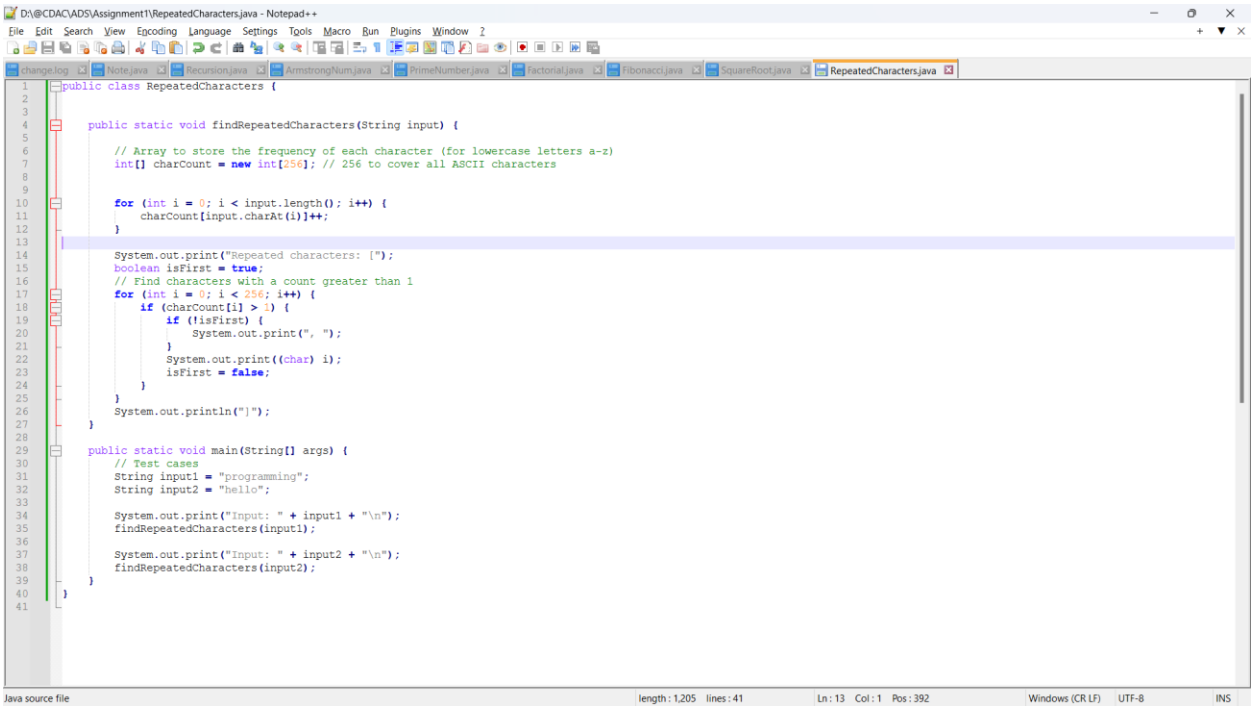
```
D:\@CDAC\ADS\Assignment1>java squareroot
Error: Could not find or load main class squareroot
Caused by: java.lang.NoClassDefFoundError: squareroot (wrong name: SquareRoot)

D:\@CDAC\ADS\Assignment1>java squareroot.java
Enter a number: 16
4

D:\@CDAC\ADS\Assignment1>java squareroot.java
Enter a number: 27
5

D:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(log x) (due to binary search)
Space Complexity: O(1) (constant space usage)

7. Find Repeated Characters in a String
Problem: Write a Java program to find all repeated characters in a string.

Test Cases:

Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']




```java
public class RepeatedCharacters {


    public static void findRepeatedCharacters(String input) {

        // Array to store the frequency of each character (for lowercase letters a-z)
        int[] charCount = new int[256]; // 256 to cover all ASCII characters


        for (int i = 0; i < input.length(); i++) {
            charCount[input.charAt(i)]++;
        }

        System.out.print("Repeated characters: [");
        boolean isFirst = true;
        // Find characters with a count greater than 1
        for (int i = 0; i < 256; i++) {
            if (charCount[i] > 1) {
                if (!isFirst) {
                    System.out.print(", ");
                }
                System.out.print((char) i);
                isFirst = false;
            }
        }
        System.out.println("]");
    }

    public static void main(String[] args) {
        // Test cases
        String input1 = "programming";
        String input2 = "hello";

        System.out.print("Input: " + input1 + "\n");
        findRepeatedCharacters(input1);

        System.out.print("Input: " + input2 + "\n");
        findRepeatedCharacters(input2);
    }
}
```

```
+-------------------------------------+
|               Start                 |
+-------------------------------------+
                |
                v
+-------------------------------------+
|         Input string from user      |
+-------------------------------------+
                |
                v
+-------------------------------------+
|  Initialize array charCount[256] to 0 |
+-------------------------------------+
                |
                v
+-------------------------------------+
|  Iterate over each character in input |
+-------------------------------------+
                |
                v
+-------------------------------------+
|   Increment count for each character |
|   in charCount[character's ASCII]    |
+-------------------------------------+
                |
                v
+-------------------------------------+
|   Check if any charCount[i] > 1      |
|   for i from 0 to 255                |
+-------------------------------------+
                |
                v
+-------------------------------------+
|  If found, print the repeated chars  |
+-------------------------------------+
                |
                v
+-------------------------------------+
|               End                   |
+-------------------------------------+
|
```

Explaination:

1 Start
- Begin the program.

2 Input: String input
- Get the string input from the user.

3 Initialize array charCount[256]
- Create an array of size 256 to store the frequency of each character (for all ASCII characters).

4 Iterate over each character of the string
- For each character in the input string, increment its corresponding count in the charCount array.

5 Check for repeated characters
- After counting, loop through the charCount array and find characters with a count greater than 1.

6 Display the repeated characters
- Print each character with a count greater than 1.

7 End

Output:

```
icrosoft Windows [Version 10.0.22631.4169]
c) Microsoft Corporation. All rights reserved.

:\@CDAC\ADS\Assignment1>javac RepeatedCharacters.java

:\@CDAC\ADS\Assignment1>java RepeatedCharacters.java
nput: programming
epeated characters: [g, m, r]
nput: hello
epeated characters: [l]

:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(n) (where n is the length of the input string)
Space Complexity: O(n) (due to storing the input string)

8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null

D:\@CDAC\ADS\Assignment1\FirstNonRepeatedCharacter.java - Notepad++

```java
public class FirstNonRepeatedCharacter {

    // Function to find the first non-repeated character
    public static Character findFirstNonRepeatedCharacter(String input) {
        int[] charCount = new int[256]; // Array to store the frequency of each character

        // First pass: Count occurrences of each character
        for (int i = 0; i < input.length(); i++) {
            charCount[input.charAt(i)]++;
        }

        // Second pass: Find the first character with a count of 1
        for (int i = 0; i < input.length(); i++) {
            if (charCount[input.charAt(i)] == 1) {
                return input.charAt(i);
            }
        }

        // If no non-repeated character is found, return null
        return null;
    }

    public static void main(String[] args) {
        // Test cases
        String input1 = "stress";
        String input2 = "aabbcc";

        Character result1 = findFirstNonRepeatedCharacter(input1);
        Character result2 = findFirstNonRepeatedCharacter(input2);

        System.out.println( input1 + "\": " + (result1 != null ? result1 : "null"));
        System.out.println( input2 + "\": " + (result2 != null ? result2 : "null"));
    }
}
```

Java source file | length : 1,242 lines : 35 | Ln : 32 Col : 28 Pos : 1,174 | Windows (CR LF) | UTF-8 | INS

```
+---------------------------------------+
|                Start                  |
+---------------------------------------+
                |
                v
+---------------------------------------+
|          Input string from user       |
+---------------------------------------+
                |
                v
+---------------------------------------+
| Initialize charCount[256] array to 0 |
+---------------------------------------+
                |
                v
+---------------------------------------+
|  First pass: Count occurrences of     |
|  each character in the string         |
+---------------------------------------+
                |
                v
+---------------------------------------+
|  Second pass: Iterate through string |
|  and find the first character whose  |
|  count in charCount array is 1        |
+---------------------------------------+
                |
     +----------+----------+
     |                     |
    Yes                   No
     |                     |
     v                     v
+------------------+   +------------------+
| Return character |   | Return null      |
+------------------+   +------------------+
                |
                v
+---------------------------------------+
|                End                    |
+---------------------------------------+
```

Explaination:

1 Start
- **Begin the program.**

2 Input: String input
- **Get the string input from the user.**

3 Initialize charCount[256] array
- **Create an array of size 256 to store the frequency of each character (for all ASCII characters).**

4 First pass: Count occurrences of each character
- **Loop through the string and increment the count of each character in the charCount array using charCount[input.charAt(i)]++.**

5 Second pass: Find the first character with count 1
- **Loop through the string again and check the charCount array to find the first character with a count of 1.**

6 Return the character or null
- **If a character with a count of 1 is found, return that character; otherwise, return null.**

7 End

```
D:\@CDAC\ADS\Assignment1>java FirstNonRepeatedCharacter.java
stress": t
aabbcc": null

D:\@CDAC\ADS\Assignment1>
```

Time Complexity: O(n)
Space Complexity: O(1)

9. Integer Palindrome
Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

Input: 121
Output: true
Input: -121
Output: false

D:\@CDAC\ADS\Assignment1\PalindromeCheck.java - Notepad++

```java
public class PalindromeCheck {

    // Function to check if an integer is a palindrome
    public static boolean isPalindrome(int number) {
        // Negative numbers are not palindromes
        if (number < 0) {
            return false;
        }

        int originalNumber = number;
        int reversedNumber = 0;

        // Reverse the number
        while (number > 0) {
            int digit = number % 10; // Get the last digit
            reversedNumber = reversedNumber * 10 + digit; // Append digit
            number /= 10; // Remove the last digit
        }

        // Check if the original number is equal to the reversed number
        return originalNumber == reversedNumber;
    }

    public static void main(String[] args) {
        // Test cases
        int input1 = 121;
        int input2 = -121;

        System.out.println(input1 +" : "+  isPalindrome(input1)); // Output: true
        System.out.println( input2  +" : "+ isPalindrome(input2)); // Output: false
    }
}
```

Java source file | length : 1,030 lines : 33 | Ln : 30 Col : 38 Pos : 973 | Windows (CR LF) UTF-8 INS

```
+-----------------------------------------+
|                Start                    |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|          Input integer number           |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|   Is number < 0?                        |
+-----------------------------------------+
     +-------------+-----------------+
     |             |                 |
    Yes            No                |
     |             |                 |
     v             v                 |
+-----------------------------------------+
|     Output: false                       |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|  Store originalNumber = number          |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
| Initialize reversedNumber = 0           |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|    While number > 0                     |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|     Get last digit: digit = number % 10 |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
| Update reversedNumber = reversedNumber * 10 + digit |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
| Update number = number / 10             |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|  Is originalNumber == reversedNumber? |
+-----------------------------------------+
     +-------------+-----------------+
     |             |                 |
    Yes            No                |
     |             |                 |
     v             v                 |
+-----------------------------------------+
|      Output: true                       |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|      Output: false                      |
+-----------------------------------------+
               |
               v
+-----------------------------------------+
|                End                      |
+-----------------------------------------+
```

Explaination:

1 Start
- Begin the program.

2 Input: Integer number
- Get the integer input from the user.

3 Check if number is negative
- If the number is negative, go to step 4; otherwise, continue to step 5.

4 Output: Return false
- If the number is negative, output false and end the program.

5 Store original number
- Store the original number for comparison later.

6 Initialize reversed number to 0
- Create a variable to hold the reversed number.

7 While number > 0
- Loop until the number becomes 0.

8 Get last digit
- Extract the last digit using number % 10.

9 Update reversed number
- Update the reversed number by appending the last digit.

10 Remove last digit from number
- Update the number by removing the last digit using integer division (number /= 10).

11 Check if original number equals reversed number
- Compare the original number with the reversed number.

12 Output result
- If they are equal, output true; otherwise, output false.

13 End
- End the program.
-

Output:

```
D:\@CDAC\ADS\Assignment1>javac PalindromeCheck.java

D:\@CDAC\ADS\Assignment1>java PalindromeCheck.java
121 : true
-121 : false
```

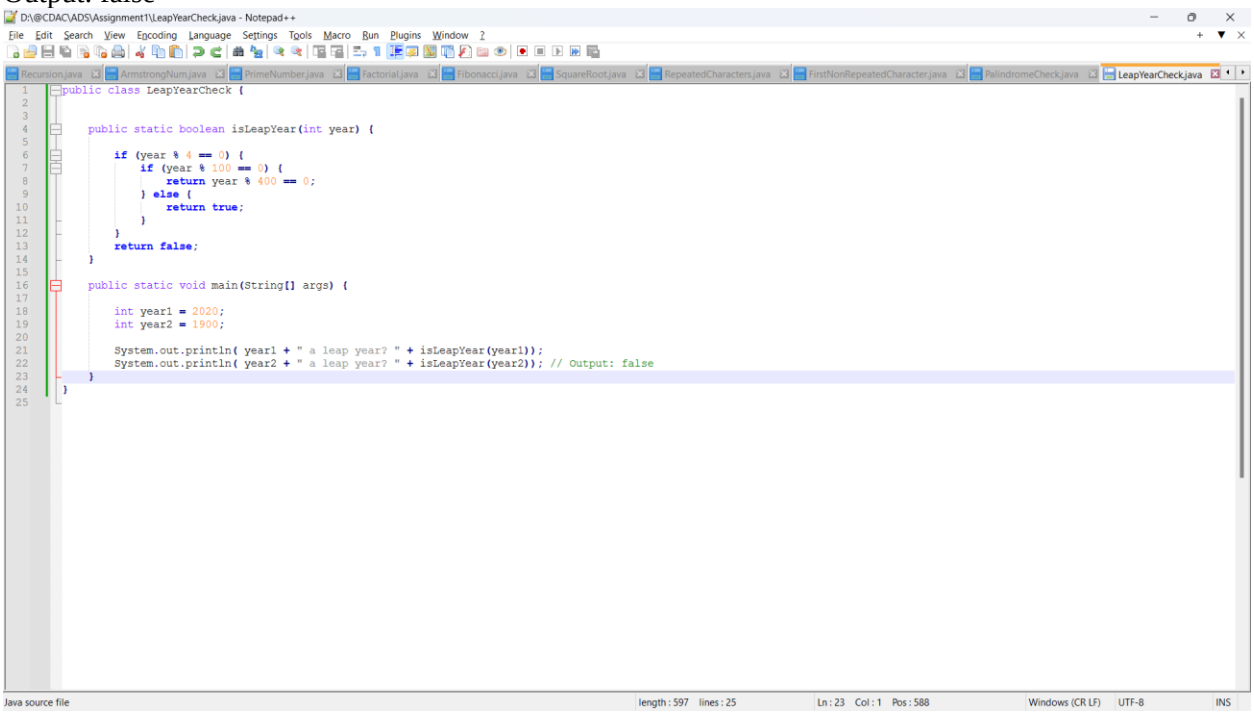
Time Complexity: **O(d)**

Space Complexity: O(1)

10. Leap Year
Problem: Write a Java program to check if a given year is a leap year.

Test Cases:

Input: 2020
Output: true
Input: 1900
Output: false




```java
public class LeapYearCheck {


    public static boolean isLeapYear(int year) {

        if (year % 4 == 0) {
            if (year % 100 == 0) {
                return year % 400 == 0;
            } else {
                return true;
            }
        }
        return false;
    }

    public static void main(String[] args) {

        int year1 = 2020;
        int year2 = 1900;

        System.out.println( year1 + " a leap year? " + isLeapYear(year1));
        System.out.println( year2 + " a leap year? " + isLeapYear(year2)); // Output: false
    }
}
```

```
+-------------------------------------+
|               Start                 |
+-------------------------------------+
               |
               v
+-------------------------------------+
|          Input year                 |
+-------------------------------------+
               |
               v
+-------------------------------------+
|   Is year % 4 == 0?                 |
+-------------------------------------+
     +------------+-----------------+
     |            |                 |
    Yes           No                |
     |            |                 |
     v            v                 |
+-------------------------------------+
|     Output: false                   |
+-------------------------------------+
               |
               v
+-------------------------------------+
|   Is year % 100 == 0?               |
+-------------------------------------+
     +------------+-----------------+
     |            |                 |
    Yes           No                |
     |            |                 |
     v            v                 |
+-------------------------------------+
|   Is year % 400 == 0?               |
+-------------------------------------+
     +------------+-----------------+
     |            |                 |
    Yes           No                |
     |            |                 |
     v            v                 |
+-------------------------------------+
|     Output: true                    |
+-------------------------------------+
               |
               v
+-------------------------------------+
|     Output: true                    |
+-------------------------------------+
               |
               v
+-------------------------------------+
|     Output: false                   |
+-------------------------------------+
               |
               v
+-------------------------------------+
|               End                   |
+-------------------------------------+
```

Explaination:

1 **Start**

- Begin the program.

2 **Input: Year**

- Get the year input from the user.

3 **Check if year is divisible by 4**

- If the year is divisible by 4, proceed to step 4; otherwise, go to step 8.

4 **Check if year is divisible by 100**

- If the year is divisible by 100, proceed to step 5; otherwise, go to step 7.

5 **Check if year is divisible by 400**

- If the year is divisible by 400, go to step 6; otherwise, go to step 8.

6 **Output: Return true**

- If the year is divisible by 400, output true (it is a leap year).

7 **Output: Return true**

- If the year is divisible by 4 but not by 100, output true (it is a leap year).

8 **Output: Return false**

- If the year is not divisible by 4, or if it is divisible by 100 but not by 400, output false (it is not a leap year).

9 **End**

- End the program.

Output:

```
D:\@CDAC\ADS\Assignment1>java FirstNonRepeatedCharacter.java
stress": t
aabbcc": null

D:\@CDAC\ADS\Assignment1>
```

Time Complexity: **O(1)**

Space Complexity**: O(1)**