ADS ASS-1

Friday, September 27, 2024 8:35 AM

C-DAC Mumbai **Date 25/09/2024**

Subject: Algorithm and Data Structure
Assignment 1

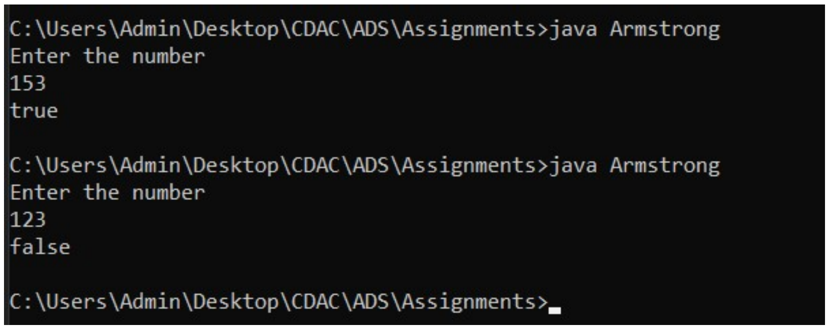
Solve the assignment with following thing to be added in each question.
-Program
-Flow chart
-Explanation
-Output
-Time and Space complexity

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

```
//Armstrong Number = It is a number that is equal to the sum of cubes of its digits.
//e.g. 153 = 1*1*1 + 5*5*5 + 3*3*3 = 1 + 125 + 27 = 153.
//

import java.util.Scanner;

class Armstrong {
        public static void main (String [] args) {
        int n;
                Scanner sc = new Scanner (System.in);
                System.out.println ("Enter the number");
                n = sc.nextInt ();
        int temp = n;       // temporary variable
        int r;
        int sum = 0;

        while (n>0) {
                r = n%10;           //to get value 3
                n = n/10;            //to remove 3 & get 15
                sum = sum + r*r*r;      // to get te cube value
        }

        if (temp == sum)
                System.out.println ("true");
        else
                System.out.println ("false");
        }
}
```



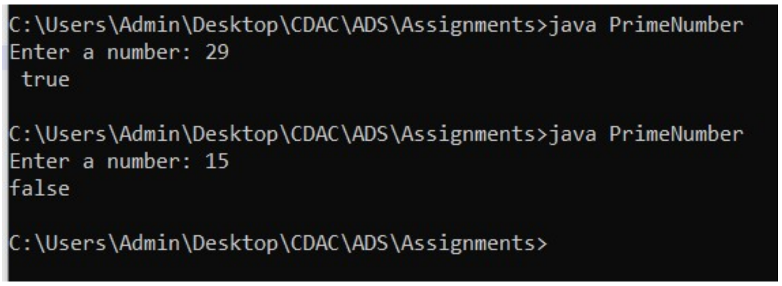
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java Armstrong
Enter the number
153
true

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java Armstrong
Enter the number
123
false

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>_
```

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

```
// Prime Number : Any number which is divisible by only 1 & itself. (0 & 1 are not prime numbers)
import java.util.Scanner;

class PrimeNumber {
    public static void main(String[] args) {
        int n;
        boolean isPrime = true; // Using boolean for clarity
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        n = sc.nextInt();

        if (n <= 1) { // check whether number is less than or equal to 1
            System.out.println(n + " is not a prime number");
        } else {
            for (int i = 2; i <= Math.sqrt(n); i++) { // Check up to the square root of n
                if (n % i == 0) {
                    isPrime = false; // Set flag to false if n is divisible
                    break; // No need to continue checking
                }
            }
            if (isPrime) {
                System.out.println(" true");
            } else {
                System.out.println("false");
            }
        }
    }
}
```

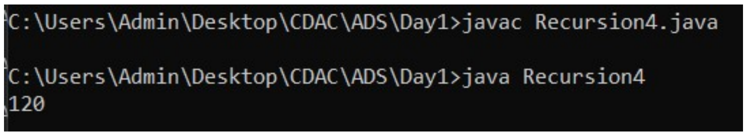

```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java PrimeNumber
Enter a number: 29
 true

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java PrimeNumber
Enter a number: 15
false

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```

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

```
// Factorial Problem :

class Recursion4 {
        int n;
                Scanner sc = new Scanner (System.in);
                System.out.println ("Enter the number");
                n = sc.nextInt ();
        static int fact (int n)
        {
                if (n<=1)
                        return 1;
                else
                        return n*fact(n-1);
        }
                public static void main (String [] args) {
                        System.out.println (fact (n));              //call for method
}
}






/* Logic for Factorial Program (Recursion Tree)

fact (5) = 5*fact(4)
       =5*4*fact(3)
         =5*4*3*fact(2)
         =5*4*3*2* fact(1)
         =5*4*3*2*1
         */
```

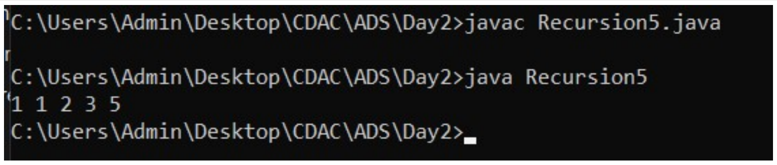

```
C:\Users\Admin\Desktop\CDAC\ADS\Day1>javac Recursion4.java

C:\Users\Admin\Desktop\CDAC\ADS\Day1>java Recursion4
120
```

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]

```
// Fibonacci Series Problem :

/*
Fib (3) = Fib (2) + Fib (1)
      = Fib (1) + Fib (0) + 1
f(n) = f(n-1) + f(n-2)
*/

class Recursion5 {
        static int fib (int n) {
           if (n <= 1) {
                return n;
        }
        return fib (n-1) + fib(n-2);

        }


        public static void main (String [] args) {
                int num = 5;
                for (int i = 1; i<=num; i++) {
                        System.out.print (fib(i)+ " ");              //call for method
                }
}
}
```



```
C:\Users\Admin\Desktop\CDAC\ADS\Day2>javac Recursion5.java

C:\Users\Admin\Desktop\CDAC\ADS\Day2>java Recursion5
1 1 2 3 5
C:\Users\Admin\Desktop\CDAC\ADS\Day2>
```

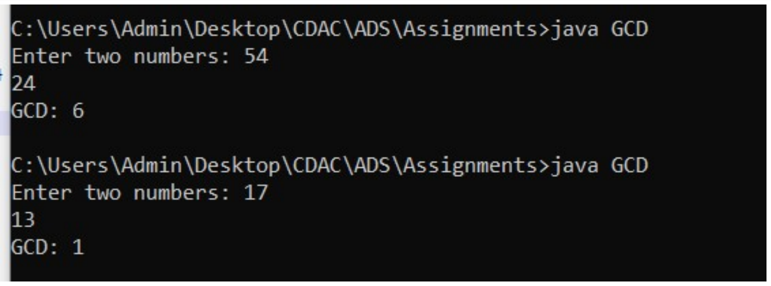
5. Find GCD
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

Test Cases:

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

```
import java.util.Scanner;

public class GCD {
   public static int gcd(int a, int b) {
      while (b != 0) {
         int temp = b;
         b = a % b;
         a = temp;
      }
```

```
        return a;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        System.out.println("GCD: " + gcd(a, b));
    }
}
```



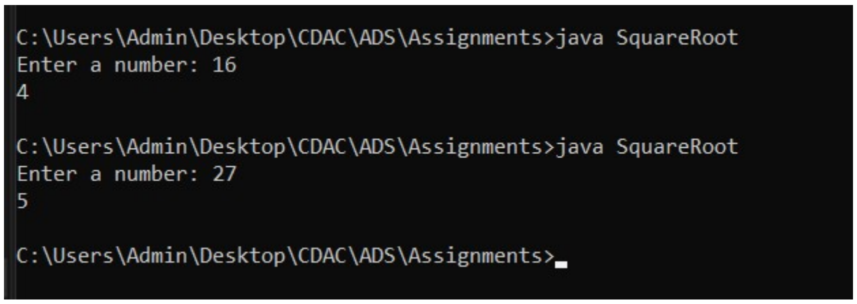
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java GCD
Enter two numbers: 54
24
GCD: 6

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java GCD
Enter two numbers: 17
13
GCD: 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

```
import java.util.Scanner;

public class SquareRoot {
    public static int sqrtRecursive(int x, int left, int right) {
        if (left > right) return right; // Base case: return right when left exceeds right
        int mid = (left + right) / 2;
        if (mid * mid == x) return mid; // Found exact square root
        if (mid * mid < x) return sqrtRecursive(x, mid + 1, right); // Search in the upper half
        return sqrtRecursive(x, left, mid - 1); // Search in the lower half
    }

    public static int sqrt(int x) {
        if (x < 0) return -1; // Invalid input
        return sqrtRecursive(x, 0, x);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int x = scanner.nextInt();
        System.out.println(sqrt(x));
    }
}
```



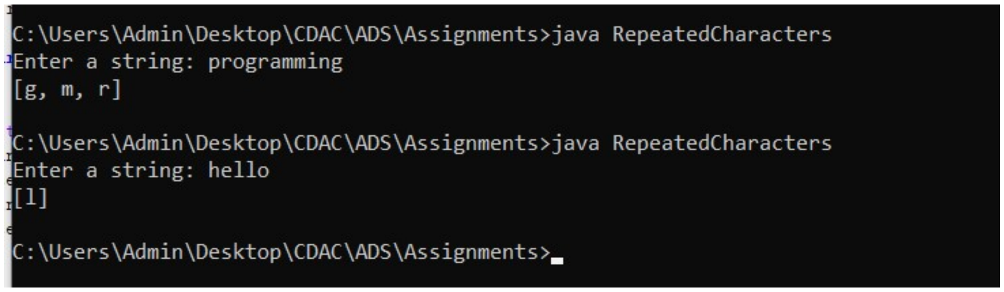
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java SquareRoot
Enter a number: 16
4

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java SquareRoot
Enter a number: 27
5

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```

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']

```
import java.util.HashMap;
import java.util.Scanner;
import java.util.ArrayList;

public class RepeatedCharacters {
    public static ArrayList<Character> findRepeated(String str) {
        HashMap<Character, Integer> charCount = new HashMap<>();
        countCharacters(str, charCount, 0);
        return getRepeatedCharacters(charCount, new ArrayList<>(), 0);
    }

    private static void countCharacters(String str, HashMap<Character, Integer> charCount, int index) {
        if (index >= str.length()) return; // Base case
        char currentChar = str.charAt(index);
        charCount.put(currentChar, charCount.getOrDefault(currentChar, 0) + 1);
        countCharacters(str, charCount, index + 1); // Recursive case
    }

    private static ArrayList<Character> getRepeatedCharacters(HashMap<Character, Integer> charCount, ArrayList<Character> result, int index) {
        if (index >= 26) return result; // Assuming only lowercase letters a-z
        char currentChar = (char) ('a' + index);
        if (charCount.getOrDefault(currentChar, 0) > 1) {
            result.add(currentChar); // Add to result if count > 1
        }
        return getRepeatedCharacters(charCount, result, index + 1); // Recursive case
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        System.out.println(findRepeated(input));
    }
}
```



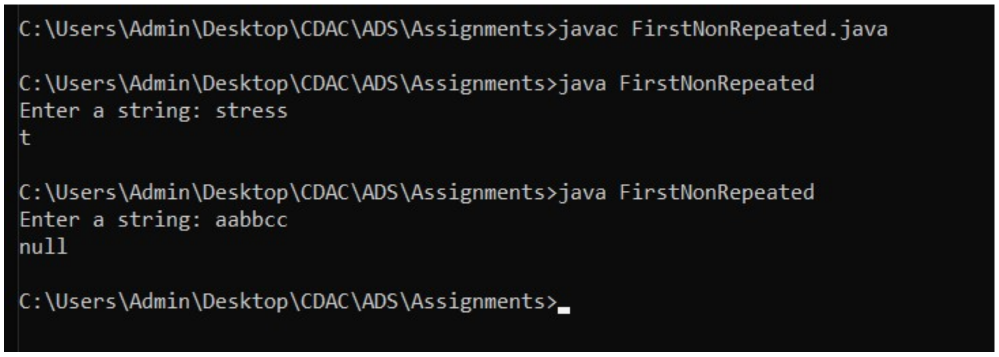
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java RepeatedCharacters
Enter a string: programming
[g, m, r]

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java RepeatedCharacters
Enter a string: hello
[l]

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```

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

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

public class FirstNonRepeated {
    public static Character firstNonRepeated(String str) {
        HashMap<Character, Integer> charCount = new HashMap<>();
        countCharacters(str, charCount, 0);
        return findFirstNonRepeated(charCount, str, 0);
    }

    private static void countCharacters(String str, HashMap<Character, Integer> charCount, int index) {
        if (index >= str.length()) return; // Base case
        char currentChar = str.charAt(index);
        charCount.put(currentChar, charCount.getOrDefault(currentChar, 0) + 1);
        countCharacters(str, charCount, index + 1); // Recursive case
    }

    private static Character findFirstNonRepeated(HashMap<Character, Integer> charCount, String str, int index) {
        if (index >= str.length()) return null; // Base case
        if (charCount.get(str.charAt(index)) == 1) {
            return str.charAt(index); // Found non-repeated character
        }
        return findFirstNonRepeated(charCount, str, index + 1); // Recursive case
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        Character result = firstNonRepeated(input);
        System.out.println(result != null ? result : "null");
    }
}
```



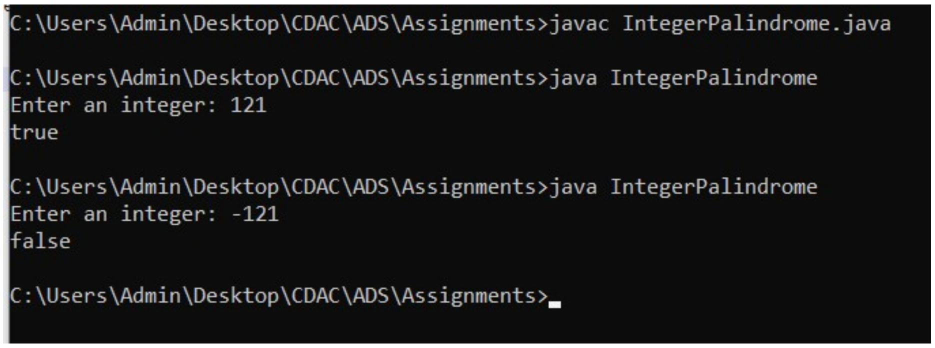
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>javac FirstNonRepeated.java

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java FirstNonRepeated
Enter a string: stress
t

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java FirstNonRepeated
Enter a string: aabbcc
null

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```

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

```
import java.util.Scanner;

public class IntegerPalindrome {
    public static boolean isPalindrome(int num) {
        if (num < 0) return false; // Negative numbers are not palindromes
        return isPalindromeHelper(num, num);
    }

    private static boolean isPalindromeHelper(int num, int original) {
        if (num == 0) return original == 0; // Base case: when the number is fully reversed
        int reversed = (int) (original % 10); // Get the last digit
        original = original / 10; // Remove the last digit from the original
        return isPalindromeHelper(num / 10, original) && (num % 10 == reversed); // Recursive case
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();
        System.out.println(isPalindrome(number));
    }
}
```



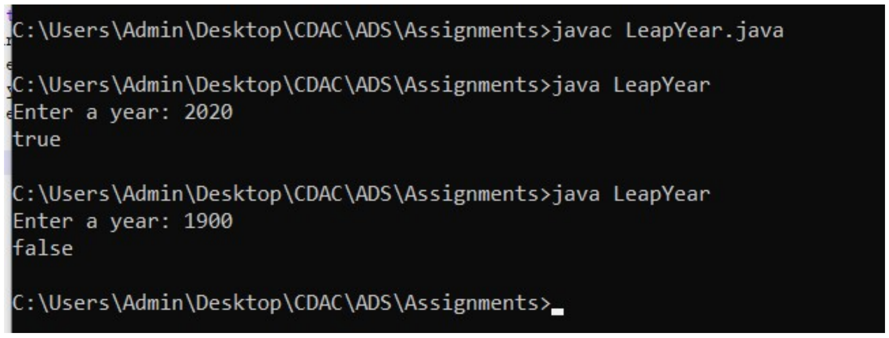
```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>javac IntegerPalindrome.java

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java IntegerPalindrome
Enter an integer: 121
true

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java IntegerPalindrome
Enter an integer: -121
false

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```

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

```
import java.util.Scanner;

public class LeapYear {
   public static boolean isLeapYear(int year) {
      return isLeapYearHelper(year);
   }

   private static boolean isLeapYearHelper(int year) {
      if (year < 0) return false; // Invalid year check
      if (year % 4 == 0) {
         if (year % 100 == 0) {
            return year % 400 == 0; // Divisible by 400
         }
         return true; // Divisible by 4 but not by 100
      }
      return false; // Not a leap year
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a year: ");
      int year = scanner.nextInt();
      System.out.println(isLeapYear(year));
   }
}
```



```
C:\Users\Admin\Desktop\CDAC\ADS\Assignments>javac LeapYear.java

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java LeapYear
Enter a year: 2020
true

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>java LeapYear
Enter a year: 1900
false

C:\Users\Admin\Desktop\CDAC\ADS\Assignments>
```