Algorithm and Data Structure

Assignment 1

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

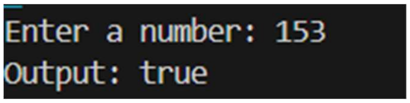
Output: false

Program:-

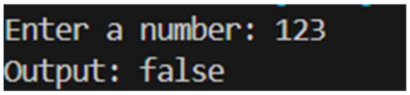
```java
package org.example;
        import java.util.Scanner;

public class ArmstrongNumber {

   // Method to check if a number is an Armstrong number
   public static boolean isArmstrong(int number) {
      int originalNumber = number;
      int sum = 0;
      int numberOfDigits = String.valueOf(number).length(); // Get number of digits
```

```
        // Calculate the sum of the digits raised to the power of number of digits
        while (number > 0) {
            int digit = number % 10;
            sum += Math.pow(digit, numberOfDigits);
            number /= 10;
        }
        // Check if the sum equals the original number
        return sum == originalNumber;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        if (isArmstrong(number)) {
            System.out.println("Output: true");
        } else {
            System.out.println("Output: false");
        }
        scanner.close();
    }
}
```

Output:



```
Enter a number: 153
Output: true
```



```
Enter a number: 123
Output: false
```

Time Complexity:- O(logN).

Space Complexity:- O(1)

2. Prime Number

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

Test Cases:

Input: 29

Output: true

Input: 15

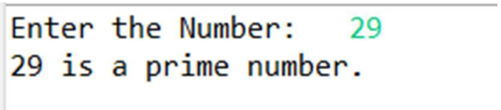
Output: false

Program:

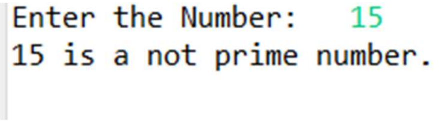
```
package org.example;
import java.util.Scanner;
public class primenumber {


    public primenumber() {
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the Number:   ");
        int num = sc.nextInt();
        boolean isPrime = true;
        if (num <= 1) {
            isPrime = false;
        } else {
```

```
      for(int i = 2; i * i <= num; ++i) {
        if (num % i == 0) {
          isPrime = false;
          break;
        }
      }
    }

    if (isPrime) {
      System.out.println("" + num + " is a prime number.");
    } else {
      System.out.println("" + num + " is a not prime number.");
    }

    sc.close();
  }
}
```

Output:



```
Enter the Number:   29
29 is a prime number.
```



```
Enter the Number:   15
15 is a not prime number.
```

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

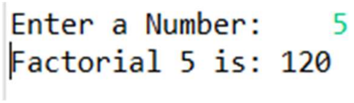
Input: 0

Output: 1

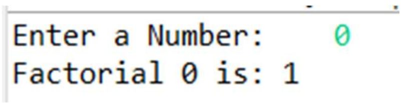
Program:

```
package org.example;
import java.util.Scanner;
public class Factorial {

public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num,result;
        result =1;
        System.out.print("Enter a Number:    ");
        num = sc.nextInt();
        for(int i = 1;i <= num; i++ ) {
                result = result*i;
        }
        System.out.println( "Factorial "+ num + " is: " + result);
        sc.close();
}
}
```

Output:



```
Enter a Number:    5
Factorial 5 is: 120
```



```
Enter a Number:    0
Factorial 0 is: 1
```

Time and Space Complexity:

Iterative Approach:

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

Recursive Approach:

- **Time Complexity: O(n)**
- **Space Complexity: O(n)**

4. Fibonacci Series

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

Test Cases:

Input: n = 5

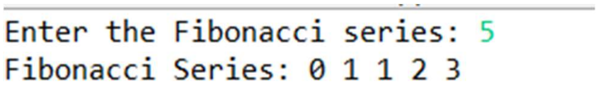
Output: [0, 1, 1, 2, 3]

Input: n = 8

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

```
package org.example;
import java.util.Scanner;
public class Fibonacci{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the Fibonacci series: ");
        int n = scanner.nextInt();

        printFibonacci(n);
        scanner.close();
    }

    public static void printFibonacci(int n) {
        int a = 0, b = 1; // First two Fibonacci numbers
        System.out.print("Fibonacci Series: ");

        for (int i = 1; i <= n; i++) {
            System.out.print(a + " "); // Print current number
            int next = a + b; // Calculate next number
            a = b; // Update a to the next number
            b = next; // Update b to the next number
        }
    }
}
```

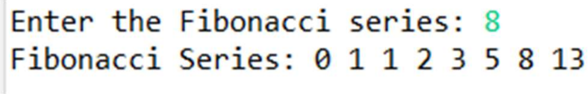
Output:



```
Enter the Fibonacci series: 5
Fibonacci Series: 0 1 1 2 3
```

```
package org.example;
import java.util.Scanner;
public class Fibonacci{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the Fibonacci series: ");
        int n = scanner.nextInt();

        printFibonacci(n);
        scanner.close();
    }

    public static void printFibonacci(int n) {
        int a = 0, b = 1; // First two Fibonacci numbers
        System.out.print("Fibonacci Series: ");

        for (int i = 1; i <= n; i++) {
            System.out.print(a + " "); // Print current number
            int next = a + b; // Calculate next number
            a = b; // Update a to the next number
            b = next; // Update b to the next number
        }
    }
}
```

Output:



```
Enter the Fibonacci series: 8
Fibonacci Series: 0 1 1 2 3 5 8 13
```

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

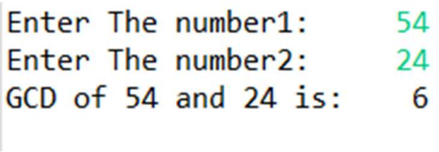
Output: 1

Program:

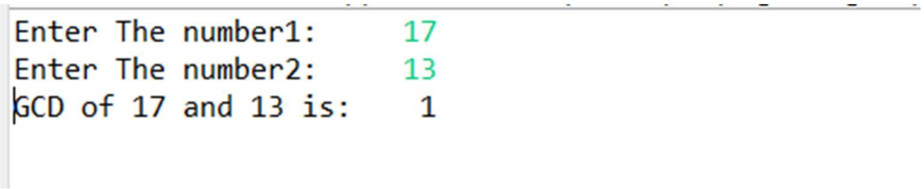
```
package org.example;
import java.util.Scanner;
public class GCD {
  public GCD() {
  }
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter The number1:     ");
    int num1 = sc.nextInt();
    System.out.print("Enter The number2:     ");
    int num2 = sc.nextInt();
    System.out.print("GCD of " + num1 + " and " + num2 + " is:    ");

    while(num2 != 0) {
      int temp = num2;
      num2 = num1 % num2;
```

```
            num1 = temp;
        }

        System.out.print(num1);
    }
}
```

Output:



```
Enter The number1:     54
Enter The number2:     24
GCD of 54 and 24 is:    6
```



```
Enter The number1:     17
Enter The number2:     13
GCD of 17 and 13 is:    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

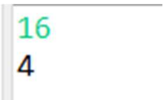
Input: x = 27

Output: 5

Program:

```
package org.example;
```

```
import java.util.Scanner;
public class squareroot {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		System.out.println((int)Math.sqrt(n));
	}
}
```



```
16
4
```

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']

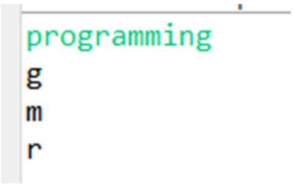
Input: "hello"

Output: ['l']

Program:

```
package org.example;
import java.util.Scanner;
public class RepeatcharString {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
```

```
                String s=sc.nextLine();
                int count[]=new int[256];
                for(char c :s.toCharArray()) {
                        count[c]++;
                }
                for(int i=0;i<count.length;i++) {
                        if(count[i]>1)
                                System.out.println((char)i+" ");
                }
}
}
```



```
programming
g
m
r
```

8. First Non-Repeated Character

Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"

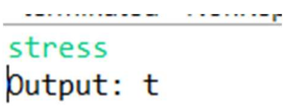
Output: 't'

Input: "aabbcc"

Output: null

Program:

```
package org.example;
import java.util.Scanner;
public class NonRepeatChar {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        char firstc='\0';
        int count[]=new int[256];
        for(char c :s.toCharArray()) {
            count[c]++;
        }
        for(char c :s.toCharArray()) {
            if(count[c]==1) {
                firstc=c;
                break;
        }
        }
         if (firstc == '\0') {
         System.out.println("Output: null");
       } else {
         System.out.println("Output: " + firstc);
       }


    }

}
```



```
stress
Output: t
```

9. Integer Palindrome

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

Test Cases:

Input: 121

Output: true

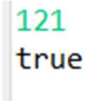
Input: -121

Output: false

Program:

```
package org.example;
import java.util.Scanner;
public class palindrome {
        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
                int n=sc.nextInt();
                int temp=n;
                int sum=0;
                if(n<0) {
                        System.out.println(false);
                        return;
                }
                while(n!=0) {
```

```
            sum=sum*10+n%10;
            n=n/10;
        }
        if(temp==sum)
            System.out.println(true);
        else
            System.out.println(false);
    }
}
```



```
121
true
```

10. Leap Year

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

Test Cases:

Input: 2020

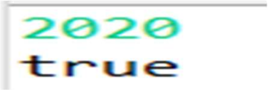
Output: true

Input: 1900

Output: false

Program:

```
package org.example;
import java.util.Scanner;
public class Leapyear {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int y=sc.nextInt();
        if((y%4==0 && y%100!=0) || y%400==0) {
            System.out.println("true");
        }
        else
            System.out.println("false");
    }
}
```



```
2020
true
```