231901020 KAVIYA V

week -01

1.1.Problem Statement:

You are required to write a Java program to calculate the total salary of an employee based on their hourly wage, hours worked in a week, and the number of weeks they worked. The program should consider the following rules:

- If an employee works more than 40 hours in a week, they are paid 1.5 times their hourly wage for the overtime hours.
- If an employee works less than 20 hours in a week, they are penalized with a deduction of 10% of their weekly salary.
- The program should handle invalid inputs (e.g., negative values for hours or wages).

Input Format:

- Hourly wage (a positive decimal value).
- Number of hours worked per week (a positive integer).
- Number of weeks worked (a positive integer).

Output Format:

Total salary considering the overtime pay and penalty rules.

SAMPLE INPUT

15.0

45

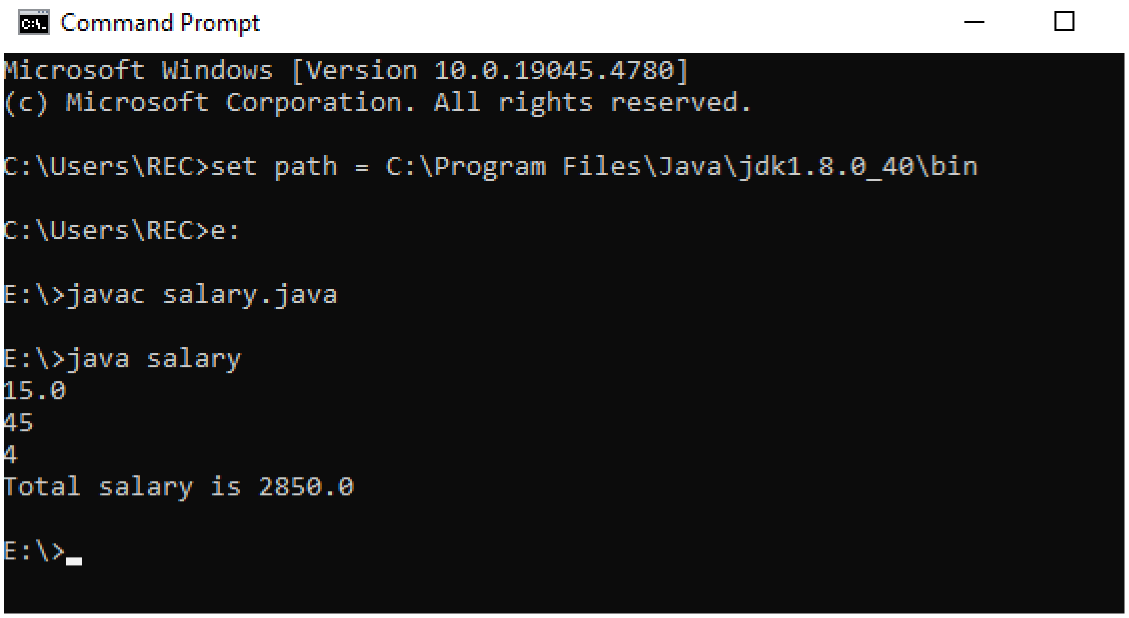
4

```
}
```

SAMPLE OUTPUT

```
Total salary is 2850.0
```

```
import java.util.Scanner;


public class salary {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double wage = sc.nextDouble();
        int hours = sc.nextInt(), weeks = sc.nextInt();


        if (wage <= 0 || hours < 0 || weeks <= 0) {
            System.out.println("Invalid input");
            return;
        }


        double totalSalary = 0.0;
        for (int i = 0; i < weeks; i++) {
            double weeklySalary = wage * (hours > 40 ? 40 + (hours - 40) * 1.5 : hours);
            if (hours < 20) weeklySalary *= 0.9;
            totalSalary += weeklySalary;
        }


        System.out.println("Total salary is " + totalSalary);
    }
}
```



Command Prompt

```
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\REC>set path = C:\Program Files\Java\jdk1.8.0_40\bin

C:\Users\REC>e:

E:\>javac salary.java

E:\>java salary
15.0
45
4
Total salary is 2850.0

E:\>
```

1.2.Problem Statement:

You are required to calculate the total cost of purchasing tickets for an event based on the ticket type and the number of tickets bought.

The program should consider the following rules:

- Regular Ticket: 50 each. If more than 10 tickets are bought, a discount of 10% is applied.
- VIP Ticket: 100 each. If more than 5 tickets are bought, a discount of 15% is applied.
- Premium Ticket: 150 each. If more than 3 tickets are bought, a discount of 20% is applied.
- If the total cost before any discount is less than 200, an additional service fee of 20 is applied.
- The program should handle invalid inputs (e.g., negative values for number of tickets, or invalid ticket types).

Input Format

Ticket type (Regular, VIP, or Premium).

Number of tickets bought (a positive integer).

Output Format

- Total cost considering the discounts and additional service fee rules

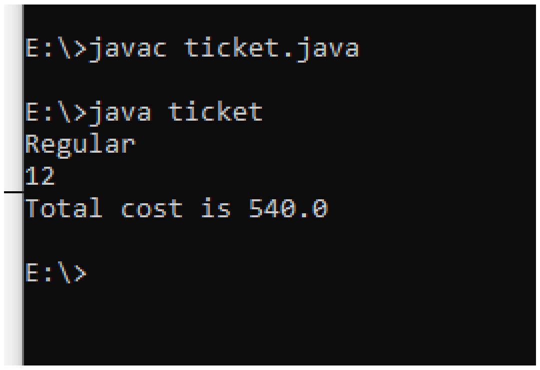
Sample Input 1

```
Regular

12
```

Sample Output 1

```
540.0
```

```
import java.util.Scanner;


public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String type = sc.next();

        int tickets = sc.nextInt();

        double cost = 0;


        if (tickets < 0 || (!type.equals("Regular") && !type.equals("VIP") &&
!type.equals("Premium"))) {

            System.out.println("Invalid input");
            return;
        }

        if (type.equals("Regular")) cost = tickets * 50 * (tickets > 10 ? 0.9 : 1);
        else if (type.equals("VIP")) cost = tickets * 100 * (tickets > 5 ? 0.85 : 1);
        else if (type.equals("Premium")) cost = tickets * 150 * (tickets > 3 ? 0.8 : 1);

        if (cost < 200) cost += 20;
        System.out.println("Total cost is " + cost);
    }
}
```



```
E:\>javac ticket.java

E:\>java ticket
Regular
12
Total cost is 540.0

E:\>
```

1.3.Largest and smallest digit of a number

Problem Statement:

Given a number N. The task is to find the largest and the smallest digit of the number.

Input Format:

A positive number in the range 1 <=n<=10000

Output Format:

Print the largest digit and the smallest digit

Sample Input

2346

Sample Output

2 6

Sample Input

4

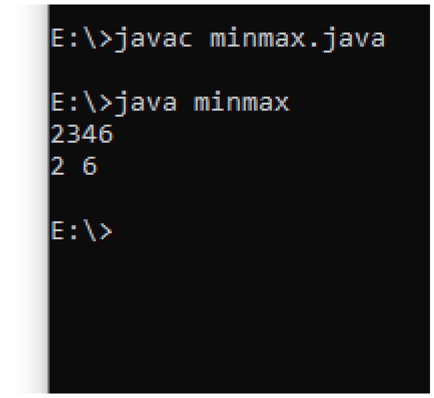
Sample Output

4 4

coding :

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt(), min = 9, max = 0;

        while (n > 0) {
            int digit = n % 10;
            if (digit > max) max = digit;
            if (digit < min) min = digit;
            n /= 10;
        }
```

```
                System.out.println(min + " " + max);
        }
}
```



```
E:\>javac minmax.java

E:\>java minmax
2346
2 6

E:\>
```

1.4.Zero-One Triangle Pattern

i) Problem Statement

This problem to understand the nested loop. Given N, a Positive integer, You are supposed to print the alternating 1’s and 0’s in triangle format.

Input Format :

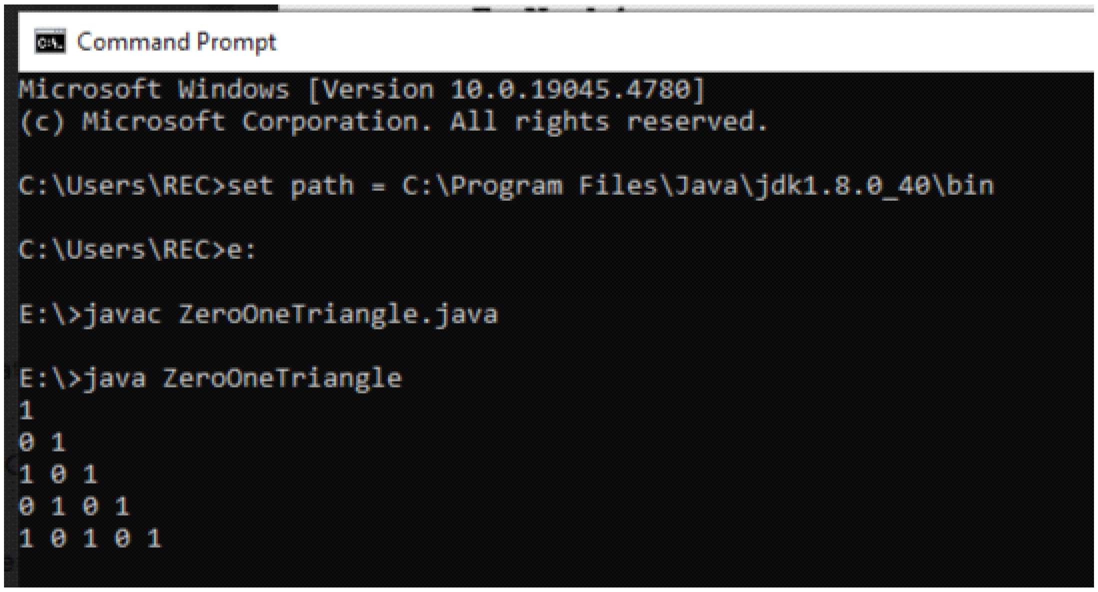
Input is positive integer : 5

Output Format:

```
1

0 1

1 0 1

0 1 0 1

1 0 1 0 1
```

coding:

```
public class ZeroOneTriangle {

    public static void main(String[] args) {

        int N = 5; // You can change N to any positive integer


        for (int i = 1; i <= N; i++) {

            for (int j = 1; j <= i; j++) {

                if ((i + j) % 2 == 0) {

                    System.out.print("1 ");

                } else {

                    System.out.print("0 ");

                }

            }

            System.out.println();

        }

    }

}
```



Command Prompt

```
Microsoft Windows [Version 10.0.19045.4780]
(c) Microsoft Corporation. All rights reserved.

C:\Users\REC>set path = C:\Program Files\Java\jdk1.8.0_40\bin

C:\Users\REC>e:

E:\>javac ZeroOneTriangle.java

E:\>java ZeroOneTriangle
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
```

ii) Number-increasing reverse Pyramid Pattern

Given N, a Positive integer, You are supposed to print in the below format.

Sample Input:

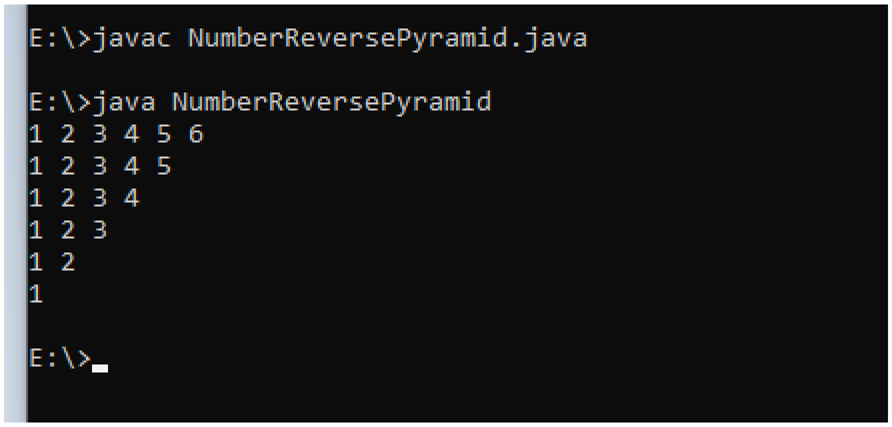
```
6
```

Sample Output:

```
1 2 3 4 5 6

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1
```

coding:

```
public class NumberReversePyramid {

    public static void main(String[] args) {

        int N = 6; // You can change N to any positive integer


        for (int i = N; i >= 1; i--) {

            for (int j = 1; j <= i; j++) {

                System.out.print(j + " ");

            }

            System.out.println();

        }

    }

}
```



```
E:\>javac NumberReversePyramid.java

E:\>java NumberReversePyramid
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

E:\>_
```

1.5.Identify the Weekday or Weekend

Problem Statement:

SYNTAX OF SWITCH CASE

The general syntax for a switch case in Java is as follows:

```
switch (expression) {
case value1:
// Code to be executed if expression equals value1
break;
case value2:
// Code to be executed if expression equals value2
break;
// ...
default:
// Code to be executed if expression doesn't match any case values
}
```

You are developing a scheduling application where users can check whether a given day is a weekday or a weekend. The application should prompt the user to enter a day of the week (e.g., "Monday", "Saturday"), and based on the input, the program should determine if the day is a weekday or a weekend.

Input Format

Input consists a week of the day

Output Format

Print whether it is weekday or weekend or invalid day

Sample Input 1

Monday

Sample Output 1

It’s a weekday

Sample Input 2

Sunday

Sample Output 2

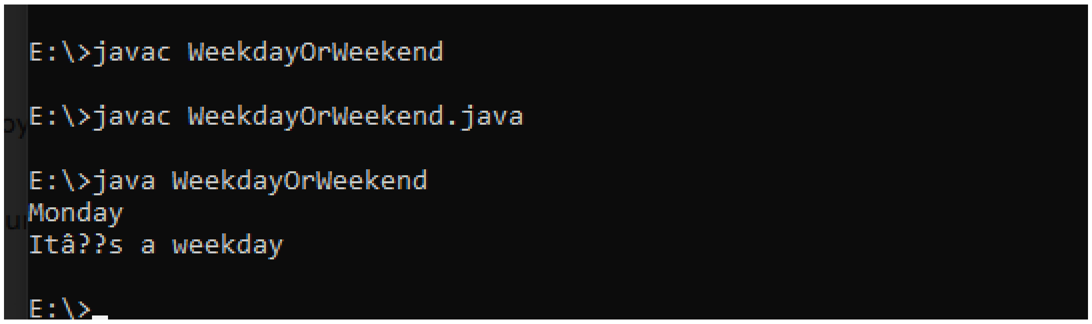
It’s a weekend

coding:

```
public class WeekdayOrWeekend {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String day = scanner.nextLine().toLowerCase();

        switch (day) {
            case "monday":
            case "tuesday":
            case "wednesday":
            case "thursday":
            case "friday":
                System.out.println("It’s a weekday");
```

```
                        break;
                case "saturday":
                case "sunday":
                        System.out.println("It’s a weekend");
                        break;
                default:
                        System.out.println("Invalid day");
        }
    }
}
```

output:



```
E:\>javac WeekdayOrWeekend

E:\>javac WeekdayOrWeekend.java

E:\>java WeekdayOrWeekend
Monday
Itâ??s a weekday

E:\>_
```

1.6.Strong Number

Problem Statement:

Write a program to check whether a number is a Strong Number or not.

A strong number is a positive integer whose sum of the factorials of its digits

equals the original number

Few examples of strong numbers are : 1,2,145 and 40585.

Input Format:

Read the positive number

Output Format:

Print Whether it is strong number or not.

Sample Input 1:

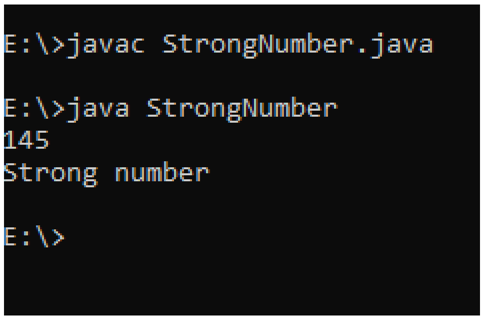
145

Sample Output 1:

Strong number

coding:

```
import java.util.Scanner;


public class StrongNumber {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int number = scanner.nextInt();

        int originalNumber = number, sum = 0;


        while (number > 0) {

            int digit = number % 10;

            int factorial = 1;

            for (int i = 1; i <= digit; i++) {

                factorial *= i;

            }

            sum += factorial;
            number /= 10;
        }

        if (sum == originalNumber) {
            System.out.println("Strong number");
        } else {
            System.out.println("Not a strong number");
        }
    }
}
```



```
E:\>javac StrongNumber.java

E:\>java StrongNumber
145
Strong number

E:\>
```