**WEEK 1**

WEEK 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 aweek,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

Program:

import java.util.Scanner;

public class Sal{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter Hourly Wages: ");

int wage = obj.nextInt();

System.out.print("Enter Hours: ");

int hours = obj.nextInt();

System.out.print("Enter Weeks: ");

int weeks = obj.nextInt();

double sal=0;

if(hours>40){

sal = (40\*wage\*weeks)+((hours-40)\*(1.5\*wage)\*weeks);

}

else if(hours<20){

sal = hours\*(wage \*(10/100))\*7\*weeks;

}

else

sal = hours\*wage\*7\*weeks;

System.out.print(sal);

}

}

WEEK 1.2: BILL GENERATION

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

PROGRAM:

import java.util.\*;

public class Bill\_Generation{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

System.out.print("Enter the Ticket Type: ");

String type = obj.nextLine().toLowerCase();

System.out.print("Enter the no. of Tickets: ");

int count = obj.nextInt();

double total=0.0;

int cost=0;

double discount=0.0;

if(type.equals("regular")){

cost = 50;

if(count>10){

discount = 0.1;

total = (cost\*count)-(cost\*count\*discount);

System.out.print(total);

}

else{

total+=20;

total = cost\*count;

System.out.print(total);

}

}

else if(type.equals("vip")){

cost = 100;

if(count>5){

discount = 0.15;

total = (cost\*count)-(cost\*count\*discount);

System.out.print(total);

}

else{

total+=20;

total = cost\*count;

System.out.print(total);

}

}

else if(type.equals("premium")){

cost = 150;

if(count>3){

discount = 0.2;

total = (cost\*count)-(cost\*count\*discount);

System.out.print(total);

}

else{

total+=20;

total = cost\*count;

System.out.print(total);

}

}

}

}

WEEK 1.3: LARGEST AND SMALLEST DIGIT IN 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

PROGRAM:

import java.util.Scanner;

public class Java\_3{

public static void main(String[] args){

Scanner obj = new Scanner(System.in);

int num = obj.nextInt();

int min=num%10,max=num%10;

int rem;

while(num!=0){

rem = num%10;

if(rem>max)

max = rem;

else if(rem<min)

min = rem;

num/=10;

}

System.out.printf("%d %d",min, max);

}

}

WEEK 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

PROGRAM:

import java.util.\*;

public class Zero\_One{

public static void main(String args[]){

Scanner obj = new Scanner(System.in);

int n = obj.nextInt();

for(int i=0; i<n; i++){

for(int j=0; j<=i;j++){

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

System.out.print("1");

else

System.out.print("0");

}

System.out.println();

}

}

}

WEEK 1.5: IDENTIFY WEEKEND OR WEEKDAY

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

PROGRAM:

import java.util.\*;

public class Week{

public static void main(String args[]){

Scanner obj = new Scanner(System.in);

System.out.print("Enter the day of the week: ");

String week = obj.nextLine();

switch(week){

case "Monday":{

System.out.print("Weekday");

break;

}

case "Tuesday":{

System.out.print("Weekday");

break;

}

case "Wednesday":{

System.out.print("Weekday");

break;

}

case "Thursday":{

System.out.print("Weekday");

break;

}

case "Friday":{

System.out.print("Weekday");

break;

}

case "Saturday":{

System.out.print("Weekend");

break;

}

case "Sunday":{

System.out.print("Weekend");

break;

}

}

}

}

WEEK 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:

PrintWhether it is strong number or not.

Sample Input 1:

145

Sample Output 1:

Strong number

PROGRAM:

import java.util.\*;

public class Strong\_Number{

public static int fact(int x){

int sum=1;

for(int i=x; i>0; i--)

sum\*=i;

return sum;

}

public static void main(String args[]){

Scanner obj = new Scanner(System.in);

int num = obj.nextInt();

int rem, sum=0;

int d = num;

while(num!=0){

rem = num%10;

sum+=fact(rem);

num/=10;

}

if(sum==d)

System.out.print("True");

else

System.out.print("False");

}

}