

## **JAVA PROGRAMS WEEK-01 AND WEEK-02:**

**Ex. No.:1.1**

### **Salary Calculation**

#### **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

#### **Program:**

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

```

public class Emp_Sal {

    public static void main(String[] args){

        double Hw, Week_sal,tot_sal;

        int Nh,Nw;

        Scanner sc = new Scanner(System.in);

        Hw = sc.nextDouble();

        Nh = sc.nextInt();

        Nw = sc.nextInt();

        if(Nh > 40){

            int Oh = Nh-40;

            Week_sal = (40*Hw)+(Oh*Hw*1.5);

            tot_sal = Week_sal*Nw;

        }

        else if(Nh < 20){

            Week_sal = Hw*Nh*(0.1);

            double Dedamt = (Hw*Nh)-Week_sal;

            tot_sal = Dedamt * Nw;

        }

        else{

            tot_sal = Hw*Nh*Nw;

        }

        System.out.println("The total salary is "+tot_sal);

    }
}

```

}

### Output:

```
D:\Java Programs>java Emp_Sal
15.0
45
4
The total salary is 2850.0
```

### Ex. No.: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.Scanner;

public class Tot_cost{

    public static void main(String[] args){

        String Tic_type;

        double Nt;

        Double Tot_cost,disc;

        Tot_cost = 0.0;

        disc = 0.0;

        Scanner sc = new Scanner(System.in);

        Tic_type = sc.nextLine();

        Nt = sc.nextDouble();


        if(Tic_type.equalsIgnoreCase("Regular")){

            if(Nt > 10){

                Tot_cost = 50*Nt*0.9;

            }

            else{
```

```

        Tot_cost = 50 *Nt;
        if(Tot_cost < 200){
            Tot_cost += 20;
        }
    }

}

else if(Tic_type.equalsIgnoreCase("VIP")){

    if(Nt > 5){
        disc = 100*Nt*1.5;
        Tot_cost = (100*Nt)-disc;
    }
    else{
        Tot_cost = 100 *Nt;
        if(Tot_cost < 200){
            Tot_cost += 20;
        }
    }

}

else if(Tic_type.equalsIgnoreCase("Premium")){

    if(Nt > 3){
        disc = 150*Nt*0.2;
        Tot_cost = (150*Nt)-disc;
    }
    else{

```

```

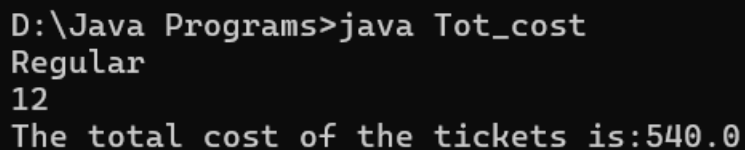
        Tot_cost = 150 *Nt;
        if(Tot_cost < 200){
            Tot_cost += 20;
        }
    }

}

if((Nt< 0) || (!Tic_type.equalsIgnoreCase("Regular") &&
!Tic_type.equalsIgnoreCase("VIP") &&
!Tic_type.equalsIgnoreCase("Premium"))){
    System.out.println("You entered an invalid ticket type or ticket number !");
}
else{
    System.out.println("The total cost of the tickets is:"+Tot_cost);
}

```

**Output:**



```

D:\Java Programs>java Tot_cost
Regular
12
The total cost of the tickets is:540.0

```

**Ex. No.: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  $1 \leq n \leq 10000$

**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 Ls {

    public static void main(String[] args){

        int n;

        Scanner sc = new Scanner(System.in);

        n = sc.nextInt();

        int t = n;

        int Large = Integer.MIN_VALUE;

        int Small = Integer.MAX_VALUE;

        while(t > 0){

            int rem = t % 10;

            if(rem > Large){

                Large = rem;

            }

            if(rem < Small){
```

```

        Small = rem;
    }
    t = t/10;
}
System.out.println(Small+" "+Large);
}
}

```

**Output:**

```

D:\Java Programs>java Ls
2346
2 6

```

**Ex. No.:1.4**

### **i) 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

```

#### **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

### **Program:**

#### **i) Zero-One Triangle Pattern**

```
import java.util.Scanner;

public class ZeroOne {

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

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

            int s = i%2;

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

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

                s = 1-s;

            }

            System.out.println();

        }

    }

}
```

### **Output:**

```
D:\Java Programs>javac ZeroOne.java
```

```
D:\Java Programs>java ZeroOne
```

```
5
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
```

## ii) Number-increasing reverse Pyramid Pattern

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

```
public class Revp {
```

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

```
        Scanner sc = new Scanner(System.in);
```

```
        int n = sc.nextInt();
```

```
        for(int i = n; i>0; i--){
```

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

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

```
            }
```

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

```
        }
```

```
    }
```

```
}
```

**Output:**

```
D:\Java Programs>javac Revp.java
D:\Java Programs>java Revp
6
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

**Ex. No.: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

### **Program:**

```
import java.util.Scanner;

public class Week{

    public static void main(String[] args){

        String Wd;

        Scanner sc = new Scanner(System.in);

        Wd = sc.nextLine();

        switch (Wd) {

            case "Monday":System.out.println("It's a weekday.");
                break;

            case "Tuesday":System.out.println("It's a weekday.");
                break;

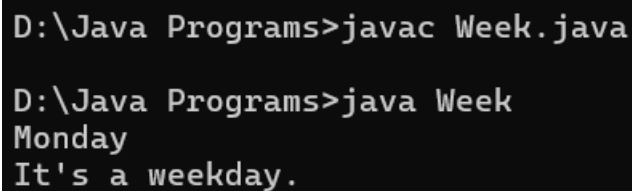
            case "Wednesday":System.out.println("It's a weekday.");
                break;

            case "Thursday":System.out.println("It's a weekday.");
                break;

            case "Friday":System.out.println("It's a weekday.");
```

```
break;
case "Saturday":System.out.println("It's a weekend.");
break;
case "Sunday":System.out.println("It's a weekdend.");
break;
    }
}
}
```

**Output:**



```
D:\Java Programs>javac Week.java
D:\Java Programs>java Week
Monday
It's a weekday.
```

**Ex. No.: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.Scanner;

public class Strong {

    public static void main(String[] args){

        int n,sum = 0;

        Scanner sc = new Scanner(System.in);

        n = sc.nextInt();

        int t = n;

        while(n > 0){

            int fact = 1;

            int rem = n%10;

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

                fact *= i;

            }

            sum += fact;

            n = n/10;

        }

        if (t == sum){

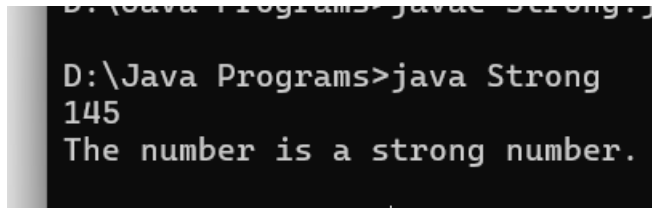
            System.out.println("The number is a strong number.");}

        else{

            System.out.println("The given number is not an Strong number.");}
```

```
    }  
    }  
}  
}
```

**Output:**



```
D:\Java Programs>java Strong  
145  
The number is a strong number.
```

**Ex. No.:2.1**

### **Count the occurrence**

#### **Problem Statement:**

You are given an array of integers nums. You are also given an integer original which is the first number that needs to be searched for in nums.

You then do the following steps:

If original is found in nums, multiply it by two (i.e., set  $\text{original} = 2 * \text{original}$ ).

Otherwise, stop the process.

Repeat this process with the new number as long as you keep finding the number.

Return the final value of original.

#### **Input Format**

First Line : The Array size n

Second Line : Array elements one in each line

Third Line : Original number

#### **Output Format**

First Line : the number

#### **Sample Input**

5

5 3 6 1 12

3

### Sample Output

24

### Explanation:

- 3 is found in nums. 3 is multiplied by 2 to obtain 6.
- 6 is found in nums. 6 is multiplied by 2 to obtain 12.
- 12 is found in nums. 12 is multiplied by 2 to obtain 24.
- 24 is not found in nums. Thus, 24 is returned.

### Program:

```
import java.util.Scanner;

public class Ar1{

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int a[] = new int[n];

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

            a[i] = sc.nextInt();

        }

        int k = sc.nextInt();

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

            if(k == a[i]){

                k *= 2;

            }

        }

        System.out.println(k);

    }

}
```

### Output:



```
D:\Java Programs>java Ar1
5
5 3 6 1 12
3
24
```

**Ex.No:2.2**

### **Inventory Management**

#### **Problem Statement:**

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Input Format

First line :The array size : 6

Second Line :the array element s : 7 1 5 3 6 4

#### **Output Format**

First Line : 5

#### **Sample Input**

6

7 1 5 3 6 4

#### **Sample Output**

5

#### **Explanation:**

Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must

buy before you sell.

**Program:**

```
import java.util.Scanner;

public class Ar2 {

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int a[] = new int[n];

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

            a[i] = sc.nextInt();

        }

        int min = a[0];

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

            if(a[i] < min){

                min = a[i];

            }

        }

        int mp = 0,p;

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

            p = a[i]-min;

            if(p > mp){

                mp = p;

            }

        }

        System.out.println(mp);

    }

}
```

```
}}
```

**Output:**

```
D:\Java Programs>java Ar2
6
7 1 5 3 6 4
5
```

**Ex. No.:2.3**

**Sort an array of 0s, 1s and 2s**

**Problem Statement:**

Given an Array of N with the elements of 0's, 1's and 2's.

Your task is to arrange the array elements in the following order.

0's followed by 1's followed 2's

**Input Format**

First Line : The Array size : n

Second Line: The array elements

**Output Format**

Print the array elements as expected.

**Sample Input 1**

6

{0, 1, 2, 0, 1, 2}

**Sample Output 1**

{0, 0, 1, 1, 2, 2}

Explanation: {0, 0, 1, 1, 2, 2} has all 0s first, then all 1s and all 2s in last.

**Input:** {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}

**Output:** {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

**Explanation:** {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2} has all 0s first, then all 1s and all 2s in last.

**Program:**

```
import java.util.Scanner;

public class Ar3 {

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int temp;

        int a[] = new int[n];

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

            a[i] = sc.nextInt();

        }

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

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

                if(a[i] <= a[j]){

                    temp = a[i];

                    a[i] = a[j];

                    a[j] = temp;

                }

            }

        }

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

            System.out.print(a[i] + " ");

        }

    }

}
```

```
}  
  
}  
  
}
```

**Output:**

```
D:\Java Programs>javac Ar3.java  
  
D:\Java Programs>java Ar3  
6  
0 1 2 0 1 2  
0 0 1 1 2 2  
D:\Java Programs>
```

**Ex.No:2.4**

### **Find the Missing Number**

#### **Problem Statement:**

Given an array `arr[]` of size `N-1` with integers in the range of `[1, N]`, the task is to find the missing number from the first `N` integers.

Note: There are no duplicates in the list.

#### **Input Format**

First Line : The array size : `N`

Second Line : The array elements

#### **Output Format**

Print the missing number .

#### **SAMPLE INPUT 1**

8

1 2 4 6 3 7 8

#### **SAMPLE OUTPUT 1**

5

Input: `arr[] = {1, 2, 4, 6, 3, 7, 8}` , `N = 8`

**Output:** 5

**Explanation:** Here the size of the array is 8, so the range will be [1, 8]

. The missing number between 1 to 8 is 5

**Input:** arr[] = {1, 2, 3, 5}, N = 5

**Output:** 4

**Explanation:** Here the size of the array is 4, so the range will be [1, 5].

The missing number between 1 to 5 is 4

**Program:**

```
import java.util.Scanner;

public class Ar4 {

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int temp;

        int a[] = new int[n];

        for(int i = 0; i < n-1; i++){

            a[i] = sc.nextInt();

        }

        for(int i = 0; i < n-1; i++){

            for(int j = 0; j < n-1; j++){

                if(a[i] <= a[j]){

                    temp = a[i];

                    a[i] = a[j];

                    a[j] = temp;

                }

            }

        }

    }

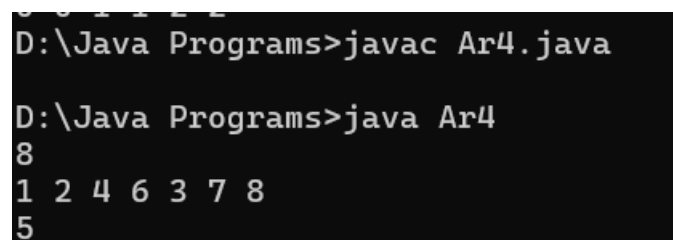
}
```

```

int k = 0;
for(int i = 0; i < n-1; i++){
    k += 1;
    if(a[i] == k){
        continue;
    }
    else{
        System.out.print(k);
        break;
    }
}
}
}
}
}

```

**Output:**



```

D:\Java Programs>javac Ar4.java
D:\Java Programs>java Ar4
8
1 2 4 6 3 7 8
5

```

**Ex. No. :2.5**

### **Move all Zeroes to the End of the Array**

#### **Problem Statement:**

Given an array of N elements, Your task is to move the Zeroes to the end of the Array.

#### **Input Format**

First Line : Array Size : N

Second Line: Array elements separated by space

## Output Format

First line: Array elements arranged as zeroes at the end.

## SAMPLE INPUT

5

1 0 2 0 3 0 4 0

## SAMPLE OUTPUT

1 2 3 4 0 0 0

## Program:

```
import java.util.Scanner;

public class Ar5 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int a[] = new int[n];

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

            a[i] = sc.nextInt();

        }

        int b[] = new int[n],k = 0;

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

            if(a[i] != 0){

                b[k] = a[i];

                k+=1;

            }

        }

        for(int i = 0; i < b.length;i++){
```



```
System.out.print(b[i] + " ");
```

```
}
```

```
}}
```

**Output:**

```
D:\Java Programs>javac Ar5.java  
  
D:\Java Programs>java Ar5  
8  
1 0 2 0 3 0 4 0  
1 2 3 4 0 0 0 0  
D:\Java Programs>|
```