IRC_SKCT_Java2_COD_Interface_abstraction

Test Summary

- No. of Sections: 1
- No. of Questions: 12
- Total Duration: 120 min

Section 1 - Coding

Section Summary

- No. of Questions: 12
- Duration: 120 min

Additional Instructions:

None

Q1. Write a java program to find the addition, subtraction, multiplication and division of two complex numbers using inheritance.
Create a **abstract class Complex**.
From this extends a child class Addition, Subtraction, Multiplication and Division.
All of these child classes should contain same methods.

Input Format

Input consists of four double type variables.
Which denotes the real and imaginary parts of the first complex number,
Followed by the real and imaginary parts of the second complex number.

Output Format

Output consists of the Addition, Subtraction, Multiplication and Division of the two input complex numbers.

Sample Input

```
2 6
4 2
```

Sample Output

```
Addition:
        6.0000 +8.0000 i
Subtraction:
        -2.0000 +4.0000 i
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. write a program to count a minimum number of front moves to sort an array.
Note: Create an interface and declare a method, the class should implement the interface.

Input Format

Input to get the size of array N in the first line, followed by N elements separated by single space in the second line.
Note: The elements must be the first N natural numbers jumbled.

Output Format

Display the output as shown in the sample output.

Constraints

N- integer type(Natural numbers)

Sample Input

```
5
2 3 1 4 5
```

Sample Output

```
1
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. Create an abstract class Shape with the following methods
abstract void rectangleArea();
abstract void squareArea();
abstract void circleArea();

Create a class Area that extends Shape that calculates and prints all the area.
Create a Main class, get the inputs and pass it to the methods.

Input Format

The first line of the input consists of the length and breadth.
The second line consists of the side.
The third line consists of the radius.

Output Format

The output prints the area of rectangle, square, and circle.
Refer sample input and output for formatting specifications.
Round off the area of the circle to two decimal places.

Sample Input

```
10 20
4
5
```

Sample Output

```
200
16
78.54
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. create an abstract class marks with the following method
void getPercentage();
Create a class A that extends marks and has 3 attributes marks1, marks2, and marks3 and a method getPercentage that calculates and prints the percentage of the student.
Create a class B that extends marks and has 4 attributes marks1, marks2, marks3, and marks4 and a method getPercentage that calculates and prints the percentage of the student.
Round off the output to two decimal places.

Input Format

The first line of the input consists of three integers i.e., the marks scored by student A.
The second line of the input consists of four integers i.e., the marks scored by student B.

Output Format

The first line prints the percentage of A.
The second line prints the percentage of B.

Sample Input

```
95 85 75
85 77 92 93
```

Sample Output

```
85.00
86.75
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. Write a program to find the sum of divisors using the concept of abstract classes.

Create an abstract class "**AbstractClass**" which contains abstract methods **getValue()** and **divisorSum(int n)**. Then write a class called Calculator which extends the abstract class.

getValue(): Method need's to get input from the user.

divisorSum(int n): Method get's "**n**" as parameter and returns the sum of the numbers divisor.

Eg: Divisor of number 4 is 1, 2, 4. Output is 1+2+4 = **7**

Input Format

The input consists of a number.

Output Format

The output prints the sum of its proper divisors.

Constraints

0 <= n <= 9999

Sample Input

```
4
```

Sample Output

```
7
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q6. Create an interface that has the following method.
void diagonalsMinMax(int a[][]);
In main method, find the smallest and largest elements in the diaginals.
Note: Only square matrix.

Input Format

Input to get the size N of the matrix in first line,the get the N*N elements in the following lines.

Output Format

Display the output as shown in the sample output.

Constraints

N -integer type.

Sample Input

```
5
7 8 9 0 1
2 3 4 5 6
5 4 2 0 8
```

Sample Output

```
Smallest Element - 1: 2
Greatest Element - 1: 32
Smallest Element - 2: 1
Greatest Element - 2: 12
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. Write a simple program to demonstrate interface.
Interface details:
Set the value of integer as 10 and then call the display function.
In display method, obtain the value of the string and display them.
In main method, display the integer value.

Input Format

The input consists of the string.

Output Format

The first line of the output prints the string.
The second line prints the integer value which is set as 10.

Constraints

Integers and strings only.

Sample Input

```
spring
```

Sample Output

```
spring
10
```

Sample Input

```
neet
```

Sample Output

```
neet
10
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. Create an interface **Produc**t with two abstract methods **void ProductDetails() and void show_Bill()**

void ProductDetails() This method will take input from the user about an order placed(no_of items, name of the individual product, and cost of the individual product and then calculate the total bill of the order using the following conditions:

if total bill is <= 10000 than, GST 0%
if the total bill is > 10000 and <=30000, then add 5 % GST to the total bill
if the total bill is >30000 and <= 50000 then add 7% GST to the total bill
if the total bill is > 50000 then add 9% GST to the total bill

void show_Bill() This method will display the total bill.

Also, create class **Customer** with two methods:
void getdetails(); This function will input details of the customer
void showdetails(); This function will print details of the customer.

Create **OrderOnline** class which is a Driver class. It will inherit the properties of the Product interface and Customer class.
Refer to the input format and output format.

Input Format

Number of customers(N)
First customer ID
First customer Name
Number of items
First Product name
Cost of the first product
Second Product name
Cost of Second product
.
.
.
Second customer ID
Second customer Name
Number of items
First Product name
Cost of the first product
.
.
.
.
Nth customer ID
Nth customer Name
Number of items
First Product name
Cost of the first product
Second Product name
Cost of Second product
.
.
.

Output Format

First customer ID
First customer name
First customer total bill
Second customer ID
Second customer name
Second customer total bill
.
.
.
.
Nth customer ID
Nth customer name
Nth customer total bill

Sample Input

```
2
618
Rahul
2
```

Sample Output

```
ID:618
NAME:Rahul
Bill:13650
ID:121
```

Sample Input

```
2
120
Radhika
3
```

Sample Output

```
ID:120
NAME:Radhika
Bill:35310
ID:151
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q9. Write a program to move all the uppercase characters to the end of the string.
Note: Create an interface with a method and the main class should define the interface method.

Input Format

Input to get a string.

Output Format

Display the output as shown in the sample output.

Sample Input

```
REasonBehInd
```

Sample Output

```
asonehndREBI
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q10. Considering the Banking Scenario, You have different types of accounts like Current Account, Savings Account which inherits the base class Account.

Create a base class **Account** with the fields - **String name, int number, int balance,** and **Date startDate**.
Create two subclasses **CurrentAccount** & **SavingsAccount** which extends **Account**.

Declare a method in Account class - calculate interest which would return the **interest** (double) and get **duedate** (Date) as a parameter. Since the Account class itself does not know how to compute calculate interest, we mark the method and class abstract.

In SavingsAccount & CurrentAccount - The interest is calculated as simple interest. (Interest 12% for the savings account and 5% for the Current account.)

Get the input from the user and print calculated interest. Refer sample input and output.

Note:
Utility methods to calculate months between two date objects.

```
import java.util.*;

 public int monthsDifference(Date startDate, Date endDate)

  Calendar c1 = new GregorianCalendar();
  c1.setTime(startDate);

  Calendar c2 = new GregorianCalendar();
  c2.setTime(endDate);

  int ans = (c2.get(c2.YEAR) - c1.get(c1.YEAR))*12;
  ans += c2.get(c2.MONTH)-c1.get(c1.MONTH);

  return ans;
}
```

Input Format

The first line of the input consists of Account Type (1 for savings account, 2 for current account).
The second line of the input consists of Name.
The third line of the input consists of Account Number.
The fourth line of the input consists of Amount.
The fifth and six-line consist of Start Date and End Date.

Output Format

The output prints the calculated Interest value.

Sample Input

```
1
Karthick
101521502
7000
```

Sample Output

```
2520.0
```

Sample Input

```
2
Karthick
111521502
4000
```

Sample Output

```
200.0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q11. Create an abstract class "Accounts" with the following attributes:
int balance;
int accno;
String name;
String address;
And the following methods
abstract public void withdrawal(int amount);
abstract public void deposit(int amount);
Display() shows the account details

Create a class SavingsAccount that extends Accounts
Attributes:
int rateOfInterest
Methods:
calcamt(int balance) that multiplies the rate of interest percentage with balance and prints the new balance.
display() that displays the account details.

deposit(int amount) that prints the new balance by adding the amount.
withdrawal(int amount) that prints the new balance by subtracting the amount.

Create a class CurrentAccount that extends Accounts
Attributes:
int draftLimit
Methods:
display() that displays the account details and draftLimit.
deposit(int amount) that prints the new balance by adding the amount.
withdrawal(int amount) that prints the new balance by subtracting the amount.

Note: consider the initial balance for deposit and withdrawal.

Input Format

The input consists of the account number, balance, account holder name and address.
The next input is choice 1 or 2 (1 for savings and 2 for current).
If the choice is 1, then the next input is the rate of interest.
The next input is the transaction type 1 or 2(1 for deposit and 2 for withdrawal)
The last input is the amount for deposit or withdrawal.

Output Format

The first line of the output prints the account details.
In the case of savings account,
Next output prints the final amount(balance*rate of interest percentage).
Next output prints the result of deposit or withdrawal (Consider the initial balance for this operation).
Refer sample input and output for formatting specifications.

Sample Input

```
252
50000
Alice
Chennai
```

Sample Output

```
252 Alice 50000 Chennai
55000
50500
```

Sample Input

```
252
50000
Alice
Chennai
```

Sample Output

```
252 Alice 50000 Chennai
60000
49500
```

Sample Input

```
888
250000
Bob
Coimbatore
```

Sample Output

```
888 Bob 250000 Coimbatore
325000
252500
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q12. Write a program, such that you should get N number of elements from the user and compute sum of elements in the odd and even position. Print the elements which has the highest sum..
Note: use interface and inheritance.

Input Format

Input to get the number of values N in first line,next line to get N number of elements followed by single space.

Output Format

Display the output as shown in the sample output.

Constraints

N (size should be even).

Sample Input

```
4
2 3 1 4
```

Sample Output

```
3 4
```

Sample Input

```
7
8 9 0 5 4 3 2
```

Sample Output

```
Enter valid number
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Coding

Q1

Test Case

Input

```
0 0
0 0
```

Output

```
Addition:
        0.0000 +0.0000 i
Subtraction:
        0.0000 +0.0000 i
```

Weightage - 10

Input

```
1 1
2 2
```

Output

```
Addition:
        3.0000 +3.0000 i
Subtraction:
        -1.0000 -1.0000 i
```

Weightage - 10

Input

```
0 3
2 7
```

Output

```
Addition:
        2.0000 +10.0000 i
Subtraction:
        -2.0000 -4.0000 i
```

Weightage - 10

Input

```
9 6
3 7
```

Output

```
Addition:
        12.0000 +13.0000 i
Subtraction:
        6.0000 -1.0000 i
```

Weightage - 10

Input

```
9 5
2 8
```

Output

```
Addition:
        11.0000 +13.0000 i
Subtraction:
        7.0000 -3.0000 i
```

Weightage - 10

Input

```
2 6
4 2
```

Output

```
Addition:
        6.0000 +8.0000 i
Subtraction:
        -2.0000 +4.0000 i
```

Weightage - 10

Input

```
3 2
7 6
```

Output

```
Addition:
        10.0000 +8.0000 i
Subtraction:
        -4.0000 -4.0000 i
```

Weightage - 10

Input

```
9 8
4 3
```

Output

```
Addition:
        13.0000 +11.0000 i
Subtraction:
        5.0000 +5.0000 i
```

Weightage - 10

Input

```
96 77
51 41
```

Output

```
Addition:
        147.0000 +118.0000 i
Subtraction:
        45.0000 +36.0000 i
```

Weightage - 10

Input

```
98 43
33 076
```

Output

```
Addition:
        131.0000 +119.0000 i
Subtraction:
        65.0000 +33.0000 i
```

Weightage - 10

Sample Input

```
2 6
4 2
```

Sample Output

```
Addition:
        6.0000 +8.0000 i
Subtraction:
        -2.0000 +4.0000 i
```

Solution

```
import java.util.*;
import java.lang.*;
import java.io.*;

abstract class Complex{
abstract float Real(float real1,float real2);
abstract float Imaginary(float imag1,float imag2);
abstract float Real1(float real1,float real2,float imag1,float imag2);
abstract float Imaginary1(float real1,float real2,float imag1,float imag2);
}
class Addition extends Complex{
float Real(float real1,float real2){
    return real1+real2;
    }
    float Imaginary(float imag1,float imag2){
    return imag1+imag2;
    }
    float Imaginary1(float real1,float real2,float imag1,float imag2){return 0;}
    float Real1(float real1,float real2,float imag1,float imag2){return 0;}
}
class Subtraction extends Complex{
float Real(float real1,float real2){
    return real1-real2;
    }
    float Imaginary(float imag1,float imag2){
    return imag1-imag2;
```

```
    }
    float Imaginary1(float real1,float real2,float imag1,float imag2){return 0;}
    float Real1(float real1,float real2,float imag1,float imag2){return 0;}
}
class Multiplication extends Complex{
float Real1(float real1,float real2,float imag1,float imag2){
    return ((real1*real2)-(imag1*imag2));
    }
float Imaginary1(float real1,float real2,float imag1,float imag2){
    return ((real1*imag2)+(imag1*real2));
    }
    float Imaginary(float imag1,float imag2){return 0;}
    float Real(float real1,float real2){return 0;}
}
class Division extends Complex{
float Real1(float real1,float real2,float imag1,float imag2){
    return (((real1*real2)+(imag1*imag2))/((real2*real2)+(imag2*imag2)));
    }
float Imaginary1(float real1,float real2,float imag1,float imag2){
    return (((imag1*real2)-(real1*imag2))/((real2*real2)+(imag2*imag2)));
    }
    float Imaginary(float imag1,float imag2){return 0;}
    float Real(float real1,float real2){return 0;}
}


class Main{
    public static void main(String args[]){
    Scanner sc =new Scanner(System.in);
    float real1=sc.nextFloat();
    float imag1=sc.nextFloat();
    float real2=sc.nextFloat();
    float imag2=sc.nextFloat();

    Complex b;
    b=new Addition();
    System.out.printf("Addition:\n\t%.4f",b.Real(real1,real2));
    if(b.Imaginary(imag1,imag2)>=0){System.out.print(" +");}
    else{System.out.print(" ");}
    System.out.printf("%.4f i",b.Imaginary(imag1,imag2));
    b=new Subtraction();
    System.out.printf("\nSubtraction:\n\t%.4f",b.Real(real1,real2));
    if(b.Imaginary(imag1,imag2)>=0){System.out.print(" +");}
    else{System.out.print(" ");}
    System.out.printf("%.4f i",b.Imaginary(imag1,imag2));
    b=new Multiplication();
    System.out.printf("\nMultiplication:\n\t%.4f",b.Real1(real1,real2,imag1,imag2));
    if(b.Imaginary1(real1,real2,imag1,imag2)>=0){System.out.print(" +");}
    else{System.out.print(" ");}
    System.out.printf("%.4f i",b.Imaginary1(real1,real2,imag1,imag2));
    b=new Division();
    System.out.printf("\nDivision:\n\t%.4f",b.Real1(real1,real2,imag1,imag2));
    if(b.Imaginary1(real1,real2,imag1,imag2)>=0){System.out.print(" +");}
    else{System.out.print(" ");}
    System.out.printf("%.4f i",b.Imaginary1(real1,real2,imag1,imag2));
}}
```

Q2

Test Case

Input	Output
7 5 6 3 2 4 1 7	4

Weightage - 15

Input

```
6
6  5  1  2  3  4
```

Output

```
5
```

Weightage - 15

Input

```
4
1  2  4  3
```

Output

```
3
```

Weightage - 15

Input

```
3
1  2  3
```

Output

```
0
```

Weightage - 25

Input

```
8
8  7  6  5  1  2  3  4
```

Output

```
7
```

Weightage - 15

Input

```
9
6  7  1  2  3  9  8  4  5
```

Output

```
8
```

Weightage - 15

Sample Input

```
5
2  3  1  4  5
```

Sample Output

```
1
```

Solution

```
import java.io.*;
import java.util.Scanner;
interface Move{
    int minmoves(int arr[], int n);
}
class Main implements Move
{
    public int minmoves(int arr[], int n)
```

```
    {
        int expectedItem = n;
        for (int i = n - 1; i >= 0; i--)
        {
            if (arr[i] == expectedItem)
                expectedItem--;
        }
          System.out.print(expectedItem);
        return expectedItem;
    }
    public static void main (String[] args)
    {
        Scanner in=new Scanner(System.in);
        int n;
        n=in.nextInt();
        int arr[] =new int[n];
        for(int i=0;i<n;i++){
            arr[i]=in.nextInt();
        }
        Main obj=new Main();
        obj.minmoves(arr,n);
}
```

Q3

Test Case

Input

```
12  24
2
3
```

Output

```
288
4
28.27
```

Weightage - 25

Input

```
18  20
6
8
```

Output

```
360
36
201.06
```

Weightage - 25

Input

```
20  24
8
7
```

Output

```
480
64
153.94
```

Weightage - 25

Input

```
28  24
10
10
```

Output

```
672
100
314.16
```

Weightage - 25

Sample Input

Sample Output

10 20 4 5	200 16 78.54

Solution

```
import java.io.*;
import java.lang.Math.*;
import java.util.*;
import java.text.*;
abstract class Shape {
    abstract void rectangleArea(int l,int b);
    abstract void squareArea(int s);
    abstract void circleArea(int r);
}
class Area extends Shape {
    public void rectangleArea(int l,int b) {
        System.out.println(l*b);
    }
    public void squareArea(int s) {
        System.out.println(s*s);
    }
    public void circleArea(int r) {
        DecimalFormat d = new DecimalFormat("0.00");
        double res = Math.PI*r*r;
        System.out.println(d.format(res));
    }
}
class Main {
    public static void main(String [] args) {
        int length,breadth,radius,side;
        Scanner sc = new Scanner(System.in);
        length = sc.nextInt();
        breadth = sc.nextInt();
        side = sc.nextInt();
        radius = sc.nextInt();
        Area a = new Area();
        a.rectangleArea(length,breadth);
        a.squareArea(side);
        a.circleArea(radius);
    }
}
```

Q4

Test Case

Input

```
95 94 98
85 86 95 98
```

Output

```
95.67
91.00
```

Weightage - 25

Input

```
99 98 97
88 87 86 85
```

Output

```
98.00
86.50
```

Weightage - 25

Input

```
95 86 74
75 65 84 50
```

Output

```
85.00
68.50
```

Weightage - 25

Input

```
90 80 60
55 66 77 88
```

Output

```
76.67
71.50
```

Weightage - 25

Sample Input

```
95 85 75
85 77 92 93
```

Sample Output

```
85.00
86.75
```

Solution

```
import java.io.*;
import java.util.*;
import java.text.*;
abstract class marks {
   abstract public void getPercentage();
}
class A extends marks {
    DecimalFormat d = new DecimalFormat("0.00");
    public int marks1;
    public int marks2;
    public int marks3;
    A() {
        this.marks1=0;
        this.marks2=0;
        this.marks3=0;
    }
    A(int m1,int m2,int m3) {
        this.marks1 = m1;
        this.marks2 = m2;
        this.marks3 = m3;
    }
    public void getPercentage() {
        int total = marks1+marks2+marks3;
        double percent = (total/300.0)*100.0;
        System.out.println(d.format(percent));
    }
}
class B extends marks {
    DecimalFormat d = new DecimalFormat("0.00");
    public int marks1;
    public int marks2;
    public int marks3;
    public int marks4;
    B() {
        this.marks1=0;
        this.marks2=0;
        this.marks3=0;
```

```
        this.marks4=0;
    }
    B(int m1,int m2,int m3,int m4) {
        this.marks1 = m1;
        this.marks2 = m2;
        this.marks3 = m3;
        this.marks4 = m4;
    }
    public void getPercentage() {
        int total = marks1+marks2+marks3+marks4;
        double percent = (total/400.0)*100.0;
        System.out.println(d.format(percent));
    }
}
class Main {
    public static void main(String args[]) {
        A a = new A();
        Scanner sc = new Scanner(System.in);
        a.marks1 = sc.nextInt();
        a.marks2 = sc.nextInt();
        a.marks3 = sc.nextInt();
        a.getPercentage();
        B b = new B();
        b.marks1 = sc.nextInt();
        b.marks2 = sc.nextInt();
        b.marks3 = sc.nextInt();
        b.marks4 = sc.nextInt();
        b.getPercentage();
    }
}
```

Q5

Test Case

Input	Output
10	18

Weightage - 10

Input	Output
13	14

Weightage - 20

Input	Output
0	0

Weightage - 20

Input	Output
9999	15912

Weightage - 20

Input	Output
100	217

Weightage - 10

Input	Output
120	360

Weightage - 10

Input	Output
333	494

Weightage - 10

Sample Input	Sample Output
4	7

Solution

```
import java.util.Scanner;

abstract class AbstractClass {
  int val;
  abstract int getValue();
  abstract int divisorSum(int n);
}

class Calculator extends AbstractClass {
public int getValue() {
    Scanner in = new Scanner(System.in);
    val = in.nextInt();
    in.close();
    return val;
}

public int divisorSum(int n) {
    int sum = 0;
    for(int i=1; i<=n; i++) {
        if(n%i == 0) {
            sum += i;
        }
    }
    return sum;
}
```

```
}

class Solution {

    public static void main(String []args) {
        Calculator calObj = new Calculator();
        int value = calObj.getValue();
        int a = calObj.divisorSum(value);
        System.out.println(a);
    }

}
```

Q6

Test Case

Input

```
6
12 32 34 45 56 78
89 90 43 56 32 16
22 24 25 62 78 90
```

Output

```
Smallest Element - 1: 12
Greatest Element - 1: 90
Smallest Element - 2: 18
Greatest Element - 2: 67
```

Weightage - 20

Input

```
4
3 4 5 6
3 2 1 0
9 7 6 2
```

Output

```
Smallest Element - 1: 2
Greatest Element - 1: 6
Smallest Element - 2: 1
Greatest Element - 2: 7
```

Weightage - 20

Input

```
3
18 19 11
21 23 17
32 12 10
```

Output

```
Smallest Element - 1: 10
Greatest Element - 1: 23
Smallest Element - 2: 11
Greatest Element - 2: 32
```

Weightage - 20

Input

```
7
12 32 34 45 56 78 43
89 90 43 56 32 16 78
22 24 25 62 78 90 22
```

Output

```
Smallest Element - 1: 12
Greatest Element - 1: 90
Smallest Element - 2: 16
Greatest Element - 2: 87
```

Weightage - 20

Input

```
10
7 8 9 0 1 8 9 3 4 5
2 3 4 5 69 9 4 5 7 6
5 4 2 0 8 0 2 1 7 6
```

Output

```
Smallest Element - 1: 2
Greatest Element - 1: 32
Smallest Element - 2: 0
Greatest Element - 2: 69
```

Weightage - 20

Sample Input

```
5
7 8 9 0 1
```

Sample Output

```
Smallest Element - 1: 2
Greatest Element - 1: 32
```

2 3 4 5 6

Smallest Element - 2: 1

Solution

```
import java.util.Scanner;
interface Matrix{
    void diagonalsMinMax(int a[][]);
}
class Main implements Matrix {
    public void diagonalsMinMax(int[][] mat)
    {
        int n = mat.length;
        if (n == 0)
           return;
        int principalMin = mat[0][0], principalMax = mat[0][0];
        int secondaryMin = mat[n-1][0], secondaryMax = mat[n-1][0];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                  if (i == j) {
                if (mat[i][j] < principalMin) {
                    principalMin = mat[i][j];
                }
          if (mat[i][j] > principalMax) {
                    principalMax = mat[i][j];
                }
          }
           if ((i + j) == (n - 1)) {
          if (mat[i][j] < secondaryMin) {
                    secondaryMin = mat[i][j];
                }
      if (mat[i][j] > secondaryMax) {
                    secondaryMax = mat[i][j];
                }
          }
    }
}

System.out.println("Smallest Element - 1: "
                   + principalMin);
System.out.println("Greatest Element - 1: "
                   + principalMax);

System.out.println("Smallest Element - 2: "
                           + secondaryMin);
        System.out.println("Greatest Element - 2: "
                           + secondaryMax);
    }
  static public void main(String[] args)
    {
        int n;
        Scanner in=new Scanner(System.in);
        n=in.nextInt();
        int matrix[][] = new int[n][n];
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
              matrix[i][j]  = in.nextInt();
            }
        }
        Main obj=new Main();
        obj.diagonalsMinMax(matrix);
    }
}
```

Q7

Test Case

Input

```
wind
```

Output

```
wind
10
```

Weightage - 10

Input

```
merging
```

Output

```
merging
10
```

Weightage - 10

Input

```
abi
```

Output

```
abi
10
```

Weightage - 10

Input

```
bala
```

Output

```
bala
10
```

Weightage - 10

Input

```
ciber
```

Output

```
ciber
10
```

Weightage - 10

Input

```
miki
```

Output

```
miki
10
```

Weightage - 10

Input

```
cool
```

Output

```
cool
10
```

Weightage - 20

Input

```
blis
```

Output

```
blis
10
```

Weightage - 20

Sample Input

```
spring
```

Sample Output

```
spring
10
```

Sample Input

```
neet
```

Sample Output

```
neet
10
```

Solution

```
import java.io.*;
import java.util.Scanner;
interface in1
{
   final int a=10;
    //Scanner sc=new Scanner(System.in);
    //a=sc.nextInt();
    void display();
}
class testClass implements in1
{
    public void display()
    {
        String sr;
        Scanner sc=new Scanner(System.in);
        sr=sc.nextLine();
        System.out.println(sr);
    }
    public static void main (String[] args)
    {
        testClass t = new testClass();
        t.display();
        System.out.println(a);
    }
}
```

Q8

Test Case

Input

```
2
250
Karan
1
```

Output

```
ID:250
NAME:Karan
Bill:5000
ID:279
```

Weightage - 20

Input

```
1
250
Karan
1
```

Output

```
ID:250
NAME:Karan
Bill:15750
```

Weightage - 20

Input

```
1
270
Arav
2
```

Output

```
ID:270
NAME:Arav
Bill:53500
```

Weightage - 20

Input

```
2
27
Aruj
1
```

Output

```
ID:27
NAME:Aruj
Bill:29400
ID:30
```

Weightage - 20

Input

```
1
279
Ravi
2
```

Output

```
ID:279
NAME:Ravi
Bill:3000
```

Weightage - 20

Sample Input

```
2
618
Rahul
2
```

Sample Output

```
ID:618
NAME:Rahul
Bill:13650
ID:121
```

Sample Input

```
2
120
Radhika
2
```

Sample Output

```
ID:120
NAME:Radhika
Bill:35310
ID:151
```

Solution

```
import java.util.*;
interface Product
{
    void ProductDetails();
    void show_Bill();
}
class Customer
{
    static Scanner sc;
    String C_Name;
    int C_ID;
    String P_Name;
    int P_Cost;
```

```
    void getdetails()
    {
       C_ID=sc.nextInt();
       sc.nextLine();
       C_Name=sc.nextLine();
    }
   void showdetails()
   {       System.out.println("ID:"+C_ID);
           System.out.println("NAME:"+C_Name);
   }}

 class OrderOnline extends Customer implements Product
{
    int no_of_items ;
    int total_Bill=0;
    public void ProductDetails()
    {
        no_of_items=sc.nextInt();

        for(int i=0;i<no_of_items;i++)
         {
              sc.nextLine();
              P_Name=sc.nextLine();
              P_Cost=sc.nextInt();
       total_Bill+=P_Cost;
      }
      if(total_Bill > 10000 && total_Bill <=30000 )
          total_Bill += (total_Bill * 5 /100 );
       else if (total_Bill > 30000 && total_Bill <=50000)
          total_Bill += (total_Bill * 7 /100 );
       else if(total_Bill > 50000)
          total_Bill += (total_Bill * 9 /100 );
}
  public void show_Bill(){
      System.out.println("Bill:"+ total_Bill);
  }
  public static void main(String []arg)
 {
     int no_of_orders;
     sc=new Scanner(System.in);
     no_of_orders=sc.nextInt();
     OrderOnline[] orders =new OrderOnline[no_of_orders];
     for(int i=0;i<no_of_orders;i++)
     {
          orders[i] =new OrderOnline();
          orders[i].getdetails();
         orders[i].ProductDetails();
        }
    for(int i=0;i<no_of_orders;i++)
       {
        orders[i].showdetails();
        orders[i].show_Bill();
        }
   }
}
```

Q9

Test Case

Input	Output
hEArmE	hrmEAE

Weightage - 15

Input

```
UnitpOlE
```

Output

```
nitplUOE
```

Weightage - 15

Input

```
rUNner
```

Output

```
rnerUN
```

Weightage - 15

Input

```
WArming
```

Output

```
rmingWA
```

Weightage - 15

Input

```
donald
```

Output

```
donald
```

Weightage - 25

Input

```
PreVieW
```

Output

```
reiePVW
```

Weightage - 15

Sample Input

```
REasonBehInd
```

Sample Output

```
asonehndREBI
```

Solution

```
import java.util.Scanner;
interface Case{
    String move(String s);
}
class Main implements Case {

    public String move(String str)
    {
```

```
    int len = str.length();
    String low = "";
    String upr = "";
   char ch;
    for (int i = 0; i < len; i++) {
        ch = str.charAt(i);
    if (ch >= 'A' && ch <= 'Z') {
            upr += ch;
        }
        else {
            low += ch;
        }
    }
    return low + upr;
}

public static void main(String args[])
{
    String str;
    Scanner in=new Scanner(System.in);
    str=in.nextLine();
    Main obj=new Main();
        obj.move(str);
        System.out.println(obj.move(str));
     }
}
```

Q10

Test Case

Input

```
1
Ram
1521502
10000
```

Output

```
2400.0
```

Weightage - 10

Input

```
1
Kumar
1521502
5000
```

Output

```
600.0
```

Weightage - 10

Input

```
1
Guru
241502
10000
```

Output

```
1200.0
```

Weightage - 10

Input

```
1
Janu
1521502
10000
```

Output

```
24000.0
```

Weightage - 15

Input

```
2
Seetha
71521502
10000
```

Output

```
500.0
```

Weightage - 15

Input

```
2
Banu
71521502
100000
```

Output

```
100000.0
```

Weightage - 20

Input

```
2
Seetha
71521502
1000
```

Output

Weightage - 20

Sample Input

```
1
Karthick
101521502
7000
```

Sample Output

```
2520.0
```

Sample Input

```
2
Karthick
111521502
4000
```

Sample Output

```
200.0
```

Solution

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Scanner;

abstract class Account{
    String name;
    int number;
    int balance;
    Date startDate;

    public Account(String name, int number, int balance, Date startDate) {
        this.name = name;
        this.number = number;
        this.balance = balance;
        this.startDate = startDate;
    }

    abstract public double calculateInterest(Date dueDate);
```

```
}

class CurrentAccount extends Account{


    public CurrentAccount(String name, int number, int balance, Date startDate) {
        super(name, number, balance, startDate);

    }

    public double calculateInterest(Date dueDate) {

        double interest;
        interest = (balance * 5 * (monthsDifference(startDate, dueDate)/12))/100;

        return interest;
    }

    public int monthsDifference(Date startDate, Date endDate) {
        Calendar c1 = new GregorianCalendar();
        c1.setTime(startDate);
        Calendar c2 = new GregorianCalendar();
        c2.setTime(endDate);
        int ans = (c2.get(c2.YEAR) - c1.get(c1.YEAR))*12;
        ans += c2.get(c2.MONTH)-c1.get(c1.MONTH);
        return ans;
    }
}

class SavingsAccount extends Account{


    public SavingsAccount(String name, int number, int balance, Date startDate) {
        super(name, number, balance, startDate);

    }

    public double calculateInterest(Date dueDate) {

        double interest;
        interest = (balance * 12 * (monthsDifference(startDate, dueDate)/12))/100;

        return interest;
    }

    public int monthsDifference(Date startDate, Date endDate) {
        Calendar c1 = new GregorianCalendar();
        c1.setTime(startDate);
        Calendar c2 = new GregorianCalendar();
        c2.setTime(endDate);
        int ans = (c2.get(c2.YEAR) - c1.get(c1.YEAR))*12;
        ans += c2.get(c2.MONTH)-c1.get(c1.MONTH);
        return ans;
    }

}

class AccountsMain{
    public static void main(String args[]) throws ParseException {
        Scanner myObj = new Scanner(System.in);

        int type=Integer.parseInt(myObj.nextLine());
        String name = myObj.nextLine();
        int number=Integer.parseInt(myObj.nextLine());
```

```
        int balance= Integer.parseInt(myObj.nextLine());

        String dateString = myObj.nextLine();
        DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
        Date startDate = formatter.parse(dateString);

        dateString = myObj.nextLine();
        Date dueDate = formatter.parse(dateString);

        if(type==1) {
            SavingsAccount sObj = new SavingsAccount(name, number, balance, startDate);
            System.out.println(sObj.calculateInterest(dueDate));
        }

        else {
            CurrentAccount cObj = new CurrentAccount(name, number, balance, startDate);
            System.out.println(cObj.calculateInterest(dueDate));
        }
        myObj.close();
    }
}
```

Q11

Test Case

Input

```
252
50000
Alice
Chennai
```

Output

```
252 Alice 50000 Chennai
60000
49500
```

Weightage - 25

Input

```
252
50000
Alice
Chennai
```

Output

```
252 Alice 50000 Chennai
55000
50500
```

Weightage - 25

Input

```
888
250000
Bob
Coimbatore
```

Output

```
888 Bob 250000 Coimbatore
325000
252500
```

Weightage - 25

Input

```
888
250000
Bob
Coimbatore
```

Output

```
888 Bob 250000 Coimbatore
325000
247500
```

Weightage - 25

Sample Input

Sample Output

```
252
50000
Alice
Chennai
```

```
252 Alice 50000 Chennai
55000
50500
```

Sample Input **Sample Output**

```
252
50000
Alice
Chennai
```

```
252 Alice 50000 Chennai
60000
49500
```

Sample Input **Sample Output**

```
888
250000
Bob
Coimbatore
```

```
888 Bob 250000 Coimbatore
325000
252500
```

Solution

```
import java.io.*;
import java.util.*;
abstract class Accounts {
    public int balance;
    public int accno;
    public String name;
    public String address;
    abstract public void withdrawl(int amount);
    abstract public void deposit(int amount);
    public Accounts(int balance,int accno,String name,String address) {
        this.balance = balance;
        this.accno = accno;
        this.name = name;
        this.address = address;
    }
    public void display() {
        System.out.println(this.accno+" "+this.name+" "+this.balance+" "+this.address);
    }
}
class SavingsAccount extends Accounts {
    int roi;
    public SavingsAccount(int balance,int accno,String name,String address) {
        super(balance,accno,name,address);
    }
public void setRoi(int r) {
    this.roi = r;
}
public int getRoi() {
    return roi;
}
public void calcamt(int balance) {
    int interestAmount = (int)balance*roi;
   // System.out.println(interestAmount);
   interestAmount /= 100;
    int finalAmount = balance+interestAmount;
    System.out.println(finalAmount);
}
public void display() {
    System.out.println(this.accno+" "+this.name+" "+this.balance+" "+this.address);
}
public void deposit(int amt) {
    int res1 = this.balance+amt;
    System.out.println(res1);
}
public void withdrawl(int amt) {
```

```
        int res2 = this.balance-amt;
        System.out.println(res2);
    }
}
class CurrentAccount extends Accounts {
    int draftLimit;
    public CurrentAccount(int balance,int accno,String name,String address) {
        super(balance,accno,name,address);
    }
    public void setDraftLimit(int dl) {
        this.draftLimit = dl;
    }
    public int getDraftLimit() {
        return draftLimit;
    }
    public void display() {
        System.out.println(this.accno+" "+this.name+" "+this.balance+" "+this.address);
        System.out.println(this.draftLimit);
    }
    public void deposit(int amt) {
        int res1 = this.balance+amt;
        System.out.println(res1);
    }
    public void withdrawl(int amt) {
        int res2 = this.balance-amt;
        System.out.println(res2);
    }
}
class Main {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        int accno = Integer.parseInt(sc.nextLine());
        int balance = Integer.parseInt(sc.nextLine());
        String name = sc.nextLine();
        String address = sc.nextLine();
        int type = Integer.parseInt(sc.nextLine());
        if(type == 1) {
            SavingsAccount s = new SavingsAccount(balance,accno,name,address);
            s.setRoi(Integer.parseInt(sc.nextLine()));
            s.display();
            s.calcamt(balance);
            int dw = Integer.parseInt(sc.nextLine());
            int amt = Integer.parseInt(sc.nextLine());
            if(dw == 1) {
                s.deposit(amt);
            }
            if(dw == 2) {
                s.withdrawl(amt);
            }
        }
        if(type == 2) {
            CurrentAccount c = new CurrentAccount(balance,accno,name,address);
            c.setDraftLimit(Integer.parseInt(sc.nextLine()));
            c.display();
            int dw1 = Integer.parseInt(sc.nextLine());
            int amt1 = Integer.parseInt(sc.nextLine());
            if(dw1 == 1) {
                c.deposit(amt1);
            }
            if(dw1 == 2) {
                c.withdrawl(amt1);
            }
        }
    }
}
```

Q12

Test Case

Input

```
6
20 30 2 2 2 10
```

Output

```
30 2 10
```

Weightage - 15

Input

```
4
3 5 2 1
```

Output

```
5 1
```

Weightage - 15

Input

```
8
2 3 4 5 8 9 0 19
```

Output

```
3 5 9 19
```

Weightage - 15

Input

```
2
34 78
```

Output

```
78
```

Weightage - 15

Input

```
10
23 25 27 28 29 32 34 37 39 30
```

Output

```
25 28 32 37 30
```

Weightage - 15

Input

```
3
4 2 1
```

Output

```
Enter valid number
```

Weightage - 25

Sample Input

```
4
2 3 1 4
```

Sample Output

```
3 4
```

Sample Input

Sample Output

```
7
8 9 0 5 4 3 2
```

```
Enter valid number
```

Solution

```
import java.util.Scanner;
interface Total{
 void printCoins(int arr[], int n) ;
}
class Coin implements Total
{
public void printCoins(int arr[], int n)
{
int oddSum = 0;
for (int i = 0; i < n; i += 2)
    oddSum += arr[i];
int evenSum = 0;
for (int i = 0; i < n; i += 2)
    evenSum += arr[i];
int start = ((oddSum > evenSum) ? 0 : 1);
for (int i = start; i < n; i += 2)
    System.out.print(arr[i]+" ");
 }}
class Main extends Coin implements Total{
public static void main(String[] args)
{
    int n1,i;
    Scanner in=new Scanner(System.in);
    n1=in.nextInt();
    int arr1[]=new int[n1];
    for(i=0;i<n1;i++){
       arr1[i]=in.nextInt();
    }
    Main obj=new Main();
    if(n1%2==0){
    obj.printCoins(arr1,n1);
    }
    else{
        System.out.print("Enter valid number");
    }
}
}
```