Experiment 1

Student Name:	**UID:**
Branch: CSE	**Section/Group:**
Semester:5	**Date of Performance:**
Subject Name:	**Subject Code: 22CSP-314**

1.1 Aim:

Given an array, of size, reverse it. Example: If array, arr= {1,4,3,2}, after reversing it, the array should be, arr= {2,3,4,1}.

2. Objective:

To understand and implement a function that reverses an array.

3. Algo/Code:

Step1 : Initialize Variables:

Create an integer array arr of the given size.

Determine the length of the array n.

Step 2:Define the Method:

Create a method reverseArray that takes the array arr as an input.

Step 3:Initialize Pointers:

Inside the method, initialize two pointers: start to 0 and end to n-1.

Step 4:Swap Elements:

Use a while loop to iterate as long as start is less than end.

Inside the loop, swap the elements at start and end positions.

Step 5:Update Pointers:

Increment start by 1 and decrement end by 1.

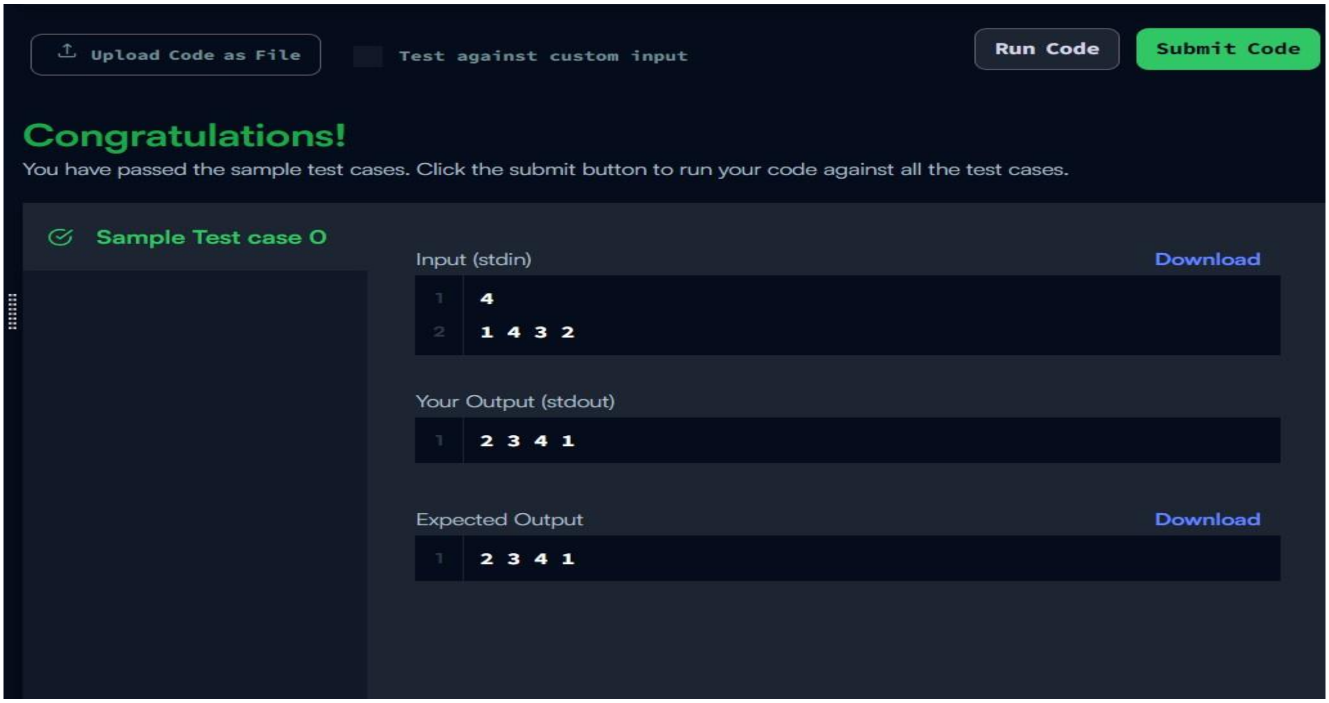
Step 6:Complete the Method:

End the while loop and the method. The array is now reversed.

Code:

```
      public class Solution {


         // Complete the reverseArray function below.
         static int[] reverseArray(int[] a) {
            for(int i = 0, j = a.length-1; i < a.length; i++, j--){
               if(i < j){
      int temp = a[i]; a[i] = a[j];
               a[j] = temp;
               }
            }
            return a;
         }
```

4. Output:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
4
1 4 3 2
```

Your Output (stdout)

```
2 3 4 1
```

Expected Output

```
2 3 4 1
```

1.2 Aim :

Given a square matrix, calculate the absolute difference between the sums of its diagonals.

For example, The square matrix is shown below:1 2 3 4 5 6 7 8 9 The left-to-right diagonal =The right to left diagonal = Their Absolute difference is.

2. Objective: solidifying the concept of array manipulation and enhancing problem solving skills in programming

3. Algo/Code:

Step 1: Initialize Variables: Create variables to hold the sums of the left-to-right and right-to-left diagonals.

Step 2: Iterate Through the Matrix: Loop through each row of the matrix.

Step 3: Sum Left-to-Right Diagonal: In each iteration, add the element from the left-to-right diagonal to its sum.

Step 4: Sum Right-to-Left Diagonal: In each iteration, add the element from the right-to-left diagonal to its sum.

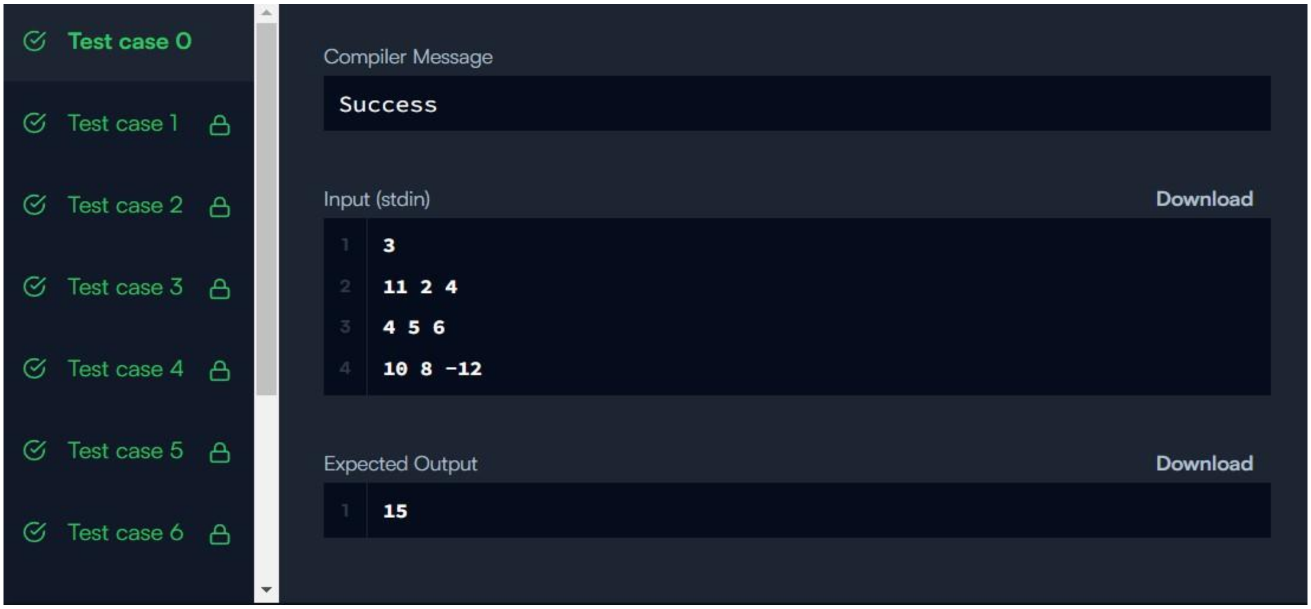
Step 5: Calculate Absolute Difference: Compute and return the absolute difference between the two sums.

Code:

```
class Result {


    /*
     * Complete the 'diagonalDifference' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts 2D_INTEGER_ARRAY arr as parameter.
```

```
 */


public static int diagonalDifference(List<List<Integer>> arr) {
// Write your code here


int rightDig = 0 ;
int leftDig = 0 ;


for( int i=0 ; i<arr.size() ; i++){


    ArrayList<Integer> al = (ArrayList)arr.get(i);
    int size = al.size();


     for(int j =0 ; j<al.size();j++){


     if(i==j){
          leftDig += al.get(j);
       }


       if(i+j==size-1){


           rightDig+= al.get(j);
       }
    }
}
return Math.abs(rightDig-leftDig);
        }
}
```

4 .Output:



Compiler Message

```
Success
```

Input (stdin)

```
3
11 2 4
4 5 6
10 8 -12
```

Expected Output

```
15
```

5. Learning Outcome

1.Both functions illustrate different aspects of algorithm design—array manipulation and matrix traversal.

2.They utilize Java syntax features like loops, conditional statements, and list manipulation (ArrayList).

3.They address common programming tasks such as array manipulation (reversal) and matrix diagonal operations (sum difference).

4.This function demonstrates how to work with nested lists (2D lists) in Java

5.It uses basic arithmetic operations and the Math.abs function to calculate and return the absolute difference.