

TASK 5

AIM: To convert a given time from **12-hour AM/PM format** into **24-hour (military) format** using Java.

ALGORITHM:

1. Read the input time string in the format hh:mm:ssAM or hh:mm:ssPM.
2. Extract:
 - The last two characters to determine AM or PM.
 - The first two characters as the hour.
 - The remaining part :mm:ss.
3. If the time is **AM**:
 - Convert 12 to 00.
4. If the time is **PM**:
 - Add 12 to the hour if it is not already 12.
5. Format the hour to always have two digits.
6. Append the minutes and seconds .
7. Return the converted time in **24-hour format**.

PROGRAM :

```
import java.io.*;

class Result {

    public static String timeConversion(String s) {

        String period = s.substring(s.length() - 2); // AM or PM

        int hour = Integer.parseInt(s.substring(0, 2));

        String rest = s.substring(2, 8); // :mm:ss

        if (period.equals("AM")) {

            if (hour == 12) {

                hour = 0;

            }

        } else { // PM

            if (hour != 12) {

                hour += 12;

            }

        }

        return String.format("%02d" + ":" + rest, hour);

    }

}
```

```

        return String.format("%02d%02s", hour, rest);

    }

public class Solution {

    public static void main(String[] args) throws IOException {

        BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(System.in)

        );

        BufferedWriter bufferedWriter = new BufferedWriter(
            new FileWriter(System.getenv("OUTPUT_PATH"))

        );

        String s = bufferedReader.readLine();

        String result = Result.timeConversion(s);

        bufferedWriter.write(result);

        bufferedWriter.newLine();

        bufferedReader.close();

        bufferedWriter.close();
    }
}

```

OUTPUT:

The screenshot shows a dark-themed interface for code submission. At the top, a green banner says "Congratulations!" followed by the text: "You have passed the sample test cases. Click the submit button to run your code against all the test cases." Below this, there are three sections: "Sample Test case 0", "Your Output (stdout)", and "Expected Output".

Section	Content
Sample Test case 0	Input (stdin) 1 07:05:45PM
Your Output (stdout)	1 19:05:45
Expected Output	1 19:05:45

Each section has a "Download" link to its right.

RESULT: The program successfully converts the given 12-hour format time into its corresponding **24-hour format**.

TASK 6

AIM: To move all zero elements in an array to the end while maintaining the relative order of the non-zero elements.

ALGORITHM;

1. Initialize an index variable to 0.
2. Traverse the array from left to right.
3. Whenever a non-zero element is found, place it at the position indicated by `index` and increment `index`.
4. After all non-zero elements are moved forward, fill the remaining positions in the array with 0.
5. Display the modified array.

PROGRAM

```
import java.util.Scanner;

class Solution {

    public void moveZeroes(int[] nums) {

        int index = 0;

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

            if (nums[i] != 0) {

                nums[index] = nums[i];

                index++;

            }

        }

        while (index < nums.length) {

            nums[index] = 0;

            index++;

        }

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of elements: ");

        int n = sc.nextInt();

        int[] nums = new int[n];

        System.out.println("Enter elements:");

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

```

        nums[i] = sc.nextInt();

    }Solution sol = new Solution();

    sol.moveZeroes(nums);

    System.out.print("Array after moving zeros: ");

    for (int num : nums) {

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

    }

    sc.close();

}

}

```

OUTPUT :

```

Accepted Runtime: 0 ms

Case 1 Case 2

Input
nums =
[0,1,0,3,12]

Output
[1,3,12,0,0]

Expected
[1,3,12,0,0]

```

RESULT: The program rearranges the array by moving all zeros to the end without changing the order of non-zero elements.

TASK 7

AIM: To calculate the **absolute difference between the sums of the primary and secondary diagonals** of a given square matrix.

ALGORITHM:

1. Read the integer n , representing the size of the square matrix.
2. Read the $n \times n$ elements of the matrix into a 2-D list.
3. Initialize two variables:
 - `primarySum` for the left-to-right diagonal.
 - `secondarySum` for the right-to-left diagonal.
4. Traverse the matrix using a loop:
 - Add `arr[i][i]` to `primarySum`.
 - Add `arr[i][n - 1 - i]` to `secondarySum`.
5. Compute the absolute difference between the two sums.
6. Display the result.

PROGRAM

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

class Result {

    public static int diagonalDifference(List<List<Integer>> arr) {

        int n = arr.size();

        int primarySum = 0;

        int secondarySum = 0;

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

            primarySum += arr.get(i).get(i);

            secondarySum += arr.get(i).get(n - 1 - i);}

        return Math.abs(primarySum - secondarySum);

    }

}

public class Solution {

    public static void main(String[] args) throws IOException {

        BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(System.in)
```

```

);

BufferedWriter bufferedWriter = new BufferedWriter(
    new FileWriter(System.getenv("OUTPUT_PATH"))

); int n = Integer.parseInt(bufferedReader.readLine().trim());

List<List<Integer>> arr = new ArrayList<>();

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

    String[] arrRowTempItems = bufferedReader.readLine().trim().split(" ");

    List<Integer> arrRowItems = new ArrayList<>();

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

        arrRowItems.add(Integer.parseInt(arrRowTempItems[j]));

    }

    arr.add(arrRowItems);

}

int result = Result.diagonalDifference(arr);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close(); }

}

```

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)		Download
1	3	
2	11 2 4	
3	4 5 6	
4	10 8 -12	

Your Output (stdout)

1	15

Expected Output

1	15

RESULT: The program successfully computes and displays the **absolute difference between the sums of both diagonals** of the given square matrix.

TASK 8

AIM: The goal is to transform a given 2D integer array (matrix) by swapping its rows and columns. Specifically, the element at position `matrix[i][j]` should move to `result[j][i]` in the new matrix.

ALGORITHM:

1. **Identify Dimensions:** Find the number of rows (`m`) and columns (`n`) of the original matrix.
2. **Initialize Result:** Create a new 2D array `result` with dimensions `n \times m` (the reverse of the original).
3. **Nested Iteration:**
4. Outer loop: Iterate through each row `i` from `0` to `m-1`.
5. Inner loop: Iterate through each column `j` from `0` to `n-1`.
6. **Value Assignment:** Map the value: `result[j][i] = matrix[i][j]`.
7. **Return:** Once all elements are processed, return the `result` matrix

PROGRAM:

```
import java.util.Arrays;

class Solution {

    public int[][] transpose(int[][] matrix) {

        int m = matrix.length;

        int n = matrix[0].length;

        int[][] result = new int[n][m]

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

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

                result[j][i] = matrix[i][j] ;}

        }

        return result;

    }

    public static void main(String[] args) {

        Solution sol = new Solution

        int[][] matrix = {

            {1, 2, 3},

            {4, 5, 6}

        }

    }

}
```

```
}

int[][] output = sol.transpose(matrix);

System.out.println("Transposed Matrix:");

for (int[] row : output) {

    System.out.println(Arrays.toString(row));}}}

}
```

OUTPUT;

The screenshot shows a user interface for testing code. At the top, there are two tabs: "Testcase" and "Test Result". The "Test Result" tab is selected, displaying the status "Accepted" and a runtime of "0 ms". Below this, there are two checkboxes: "Case 1" (which is checked) and "Case 2" (which is also checked). The "Input" section contains the following Java code:

```
matrix =
[[1,2,3],[4,5,6],[7,8,9]]
```

The "Output" section shows the transposed matrix:

```
[[1,4,7],[2,5,8],[3,6,9]]
```

The "Expected" section also shows the same transposed matrix:

```
[[1,4,7],[2,5,8],[3,6,9]]
```

RESULT ; The program has been executed successfully, and the resulting transposed matrix is generated as a new two-dimensional array where the original rows have been converted into columns.

