DAY 4 PRACTICE QUESTIONS

1.Manage the scores of students:

Scenario: Write a program to manage the scores of students in a test. The program should:

1. Store the scores of 5 students in an array.
2. Calculate the average score.
3. Find the highest score.
4. Find the lowest score.
5. Print all the scores along with the calculated average, highest, and lowest scores.

Requirements:

- Use a single-dimensional array to store the scores.
- Use loops to iterate through the array for calculations.

*****Test Cases*****

Test Case 1: Standard Set of Scores

Input Scores: {85, 92, 78, 90, 88}

Expected Output:

Scores: 85 92 78 90 88

Average Score: 86.6

Highest Score: 92

Lowest Score: 78

Test Case 2: Scores Including Maximum Value

Input Scores: {70, 80, 90, 100, 60}

Expected Output:

Scores: 70 80 90 100 60

Average Score: 80.0

Highest Score: 100

Lowest Score: 60

Test Case 3: Incremental Scores

Input Scores: {55, 65, 75, 85, 95}

Expected Output:

Scores: 55 65 75 85 95

Average Score: 75.0

Highest Score: 95

Lowest Score: 55

Test Case 4: All Scores Identical

Input Scores: {100, 100, 100, 100, 100}

Expected Output:

Scores: 100 100 100 100 100

Average Score: 100.0

Highest Score: 100

Lowest Score: 100

Test Case 5: Scores Including Minimum Value

Input Scores: {45, 55, 65, 35, 25}

Expected Output:

Scores: 45 55 65 35 25

Average Score: 45.0

Highest Score: 65

Lowest Score: 25

2. Manage a list of names for a small event

Scenario: write a program to manage a list of names for a small event. The program should:

1. Store the names of 5 attendees in an array.
2. Find and print the longest name.
3. Find and print the shortest name.
4. Print all the names in reverse order.

Requirements:

- Use a single-dimensional array to store the names.
- Use loops to iterate through the array for finding the longest and shortest names and for printing names in reverse order.

*****Test Cases*****

Test Case 1: Standard Set of Names

Input Names: {"Alice", "Bob", "Charlotte", "David", "Eve"}

Expected Output:

Names in Reverse Order: Eve, David, Charlotte, Bob, Alice

Longest Name: Charlotte

Shortest Name: Bob

Test Case 2: Names with Same Lengths

Input Names: {"Tom", "Sam", "Jim", "Tim", "Kim"}

Expected Output:

Names in Reverse Order: Kim, Tim, Jim, Sam, Tom

Longest Name: Tom

Shortest Name: Tom

Test Case 3: Names with Mixed Lengths

Input Names: {"Ann", "Elizabeth", "Joe", "Alex", "Zoe"}

Expected Output:

Names in Reverse Order: Zoe, Alex, Joe, Elizabeth, Ann

Longest Name: Elizabeth

Shortest Name: Ann

Test Case 4: Names with Varying Cases

Input Names: {"alice", "Bob", "CHARLOTTE", "david", "EVE"}

Expected Output:

Names in Reverse Order: EVE, david, CHARLOTTE, Bob, alice

Longest Name: CHARLOTTE

Shortest Name: Bob

Test Case 5: Single Character Names

Input Names: {"A", "B", "C", "D", "E"}

Expected Output:

Names in Reverse Order: E, D, C, B, A

Longest Name: A

Shortest Name: A

3. Manage Grades Of Students

Scenario: You are a teacher who needs to keep track of the grades of students in two subjects: Math and Science. You decide to use a two-dimensional array to store the grades of 5 students. Each row in the array represents a student, and the first column contains the Math grade, while the second column contains the Science grade.

Requirements:

1. Create a two-dimensional array to store the grades.

   Initialize the array with the following grades:

- Student 1: Math: 85, Science: 92
- Student 2: Math: 78, Science: 85
- Student 3: Math: 90, Science: 88
- Student 4: Math: 95, Science: 94
- Student 5: Math: 80, Science: 87

2. Write a method to print the grades of all students.
3. Write a method to calculate and print the average grade for each subject.

*****Test Cases*****

Test Case 1: Standard Grades

Description: Test the program with the provided standard grades for 5 students.

Input:

```
int[][] grades = {
   {85, 92},
   {78, 85},
   {90, 88},
   {95, 94},
   {80, 87}
};
```

Expected Output:

```
Grades of all students:
Student 1: Math: 85, Science: 92
Student 2: Math: 78, Science: 85
Student 3: Math: 90, Science: 88
Student 4: Math: 95, Science: 94
Student 5: Math: 80, Science: 87
Average Math grade: 85.6
Average Science grade: 89.2
```

Test Case 2: All Perfect Scores

Description: Test the program with perfect scores (100) for all students in both subjects.

Input:

```
int[][] grades = {
   {100, 100},
   {100, 100},
   {100, 100},
   {100, 100},
   {100, 100}
};
```

Expected Output:

```
Grades of all students:
Student 1: Math: 100, Science: 100
Student 2: Math: 100, Science: 100
Student 3: Math: 100, Science: 100
Student 4: Math: 100, Science: 100
Student 5: Math: 100, Science: 100
Average Math grade: 100.0
Average Science grade: 100.0
```

Test Case 3: All Failing Scores

Description: Test the program with failing scores (below 60) for all students in both subjects.

Input:

```
int[][] grades = {
    {50, 55},
    {45, 58},
    {55, 52},
    {48, 49},
    {54, 51}
};
```

Expected Output:

```
Grades of all students:
Student 1: Math: 50, Science: 55
Student 2: Math: 45, Science: 58
Student 3: Math: 55, Science: 52
Student 4: Math: 48, Science: 49
Student 5: Math: 54, Science: 51
Average Math grade: 50.4
Average Science grade: 53.0
```

Test Case 4: Mixed Grades

Description: Test the program with a mix of high and low scores for different students.

Input:

```
int[][] grades = {
    {70, 95},
    {85, 40},
    {60, 78},
    {92, 88},
    {73, 55}
};
```

Expected Output:

```
Grades of all students:
Student 1: Math: 70, Science: 95
Student 2: Math: 85, Science: 40
Student 3: Math: 60, Science: 78
Student 4: Math: 92, Science: 88
Student 5: Math: 73, Science: 55
Average Math grade: 76.0
Average Science grade: 71.2
```

Test Case 5: Edge Case with Zero Grades

Description: Test the program with all zero scores for all students in both subjects.

```
int[][] grades = {
    {0, 0},
    {0, 0},
    {0, 0},
    {0, 0},
    {0, 0}
};
```

Expected Output:

```
Grades of all students:
Student 1: Math: 0, Science: 0
Student 2: Math: 0, Science: 0
Student 3: Math: 0, Science: 0
Student 4: Math: 0, Science: 0
Student 5: Math: 0, Science: 0
Average Math grade: 0.0
```

Average Science grade: 0.0

4. Manage Grades of Students

Scenario: You are a teacher at a school and you want to keep track of the scores of students in different subjects. You decide to use a multi-dimensional array to store the scores. The school has 3 students and each student has scores in 4 subjects: Math, Science, English, and History.

Requirements:

1. Create a 2D array to store the scores of 3 students in 4 subjects.
2. Initialize the array with some sample scores.
3. Write a method to calculate the average score for each student.
4. Write a method to calculate the average score for each subject.
5. Print the average scores for each student and each subject.

*****Test Cases*****

Test Case 1: Average Calculation with Uniform Scores

Description: Input consists of uniform scores across subjects for all students. Expected output verifies correct calculation of average scores for each student and subject.

Input:

```
int[][] scores = {

    {80, 80, 80, 80},

    {80, 80, 80, 80},

    {80, 80, 80, 80}

};
```

Expected Output:

```
Average scores for each student:

Student 1: 80.0

Student 2: 80.0

Student 3: 80.0

Average scores for each subject:

Math: 80.0

Science: 80.0

English: 80.0

History: 80.0
```

Test Case 2: Average Calculation with Varying Student Scores

Description: Input contains varying scores across subjects for different students. Expected output validates accurate calculation of average scores for each student and subject.

Input:

```
int[][] scores = {
    {85, 78, 90, 88},
    {76, 85, 83, 80},
    {91, 89, 92, 94}
};
```

Expected Output:

```
Average scores for each student:
Student 1: 85.25
Student 2: 81.0
Student 3: 91.5
Average scores for each subject:
Math: 84.0
Science: 84.0
English: 88.33333333333333
History: 87.33333333333333
```

Test Case 3: Empty Array

Description: Input consists of an empty array. Expected output ensures graceful handling of empty input scenario without errors.

Input:

```
int[][] scores = {};
```

Expected Output:

```
Average scores for each student:
Average scores for each subject:
```

Test Case 4: Single Student with Various Scores

Description: Input includes scores for a single student across different subjects. Expected output verifies correct calculation of average scores for the single student and each subject.

Input:

```
int[][] scores = {
    {70, 85, 90, 75}
};
```

Expected Output:

```
Average scores for each student:
Student 1: 80.0
```

```
Average scores for each subject:
Math: 70.0
Science: 85.0
English: 90.0
History: 75.0
```

Test Case 5: Large Dataset

Description: Input consists of scores for multiple students and subjects. Expected output validates efficient calculation of average scores for each student and subject in a larger dataset scenario.

Input:

```
int[][] scores = {
    {75, 80, 85, 90},
    {82, 88, 76, 90},
    {90, 92, 87, 85},
    {78, 85, 80, 88},
    {85, 79, 91, 84}
};
```

Expected Output:

```
Average scores for each student:
Student 1: 82.5
Student 2: 84.0
Student 3: 88.5
Student 4: 82.75
Student 5: 84.75
Average scores for each subject:
Math: 82.0
Science: 84.8
English: 83.8
History: 87.4
```

5. Greetings

Scenario:Imagine you're a teacher in a classroom. You have a list of students' names and you want to greet each student individually.

Requirements: Use a foreach loop to greet each student by their name.

*****Test Cases*****

Test Case 1: All Students Greeted:

Description: Ensure all students in the list are greeted individually.

- Input: List of student names: ["Alice", "Bob", "Charlie", "David", "Eve"]
- Expected Output: Five lines printed, each greeting a different student.

Test Case 2:Empty List Handling:

Description: Test behavior when the list of students is empty.

- Input: Empty list: []
- Expected Output: No output (program terminates without printing anything).

Test Case 3:Single Student Case:

Description: Test behavior when there is only one student in the list.

- Input: List with one student: ["John"]
- Expected Output: One line printed, greeting the single student.

Test Case 4: Special Characters:

Description: Test behavior when student names contain special characters or numbers.

- Input: List with names containing special characters: ["Anna-Maria", "Joe123", "Sam@Home"]
- Expected Output: Three lines printed, each greeting a different student with special characters.

Test Case 5: Performance Test:

Description: Test the performance with a large list of students.

- Input: List with a large number of students (e.g., 1000 names).
- Expected Output: 1000 lines printed, each greeting a different student.