Project
CPE 593

Assignment Title: Student Exam Score Analyzer with Bubble Sort and Binary Search

Description:

In this programming assignment, you will develop a C++ program that performs a wide range of tasks related to student exam scores. Your program will not only analyze student exam scores and calculate statistics but also incorporate the **Bubble Sort algorithm** for sorting scores and the **Binary Search algorithm** for efficient searching. Additionally, the program will assign grades and provide a visual representation of the exam score distribution in the form of a histogram.

Task 1: Input Student Details and Sort Exam Scores (Bubble Sort)

- Function Signature: `void inputStudentDetailsAndSort(string names[], int scores[][MAX_EXAMS], int numStudents, int numExams)`
- Description: This function will prompt the user to input details for multiple students, including their names and exam scores. It will also sort the exam scores for each student in ascending order using the Bubble Sort algorithm.
- Input:
  - `names[]`: An array of strings to store student names.
  - `scores[][MAX_EXAMS]`: A 2D array to store exam scores for each student.
  - `numStudents`: The number of students.
  - `numExams`: The number of exams per student.

Task 2: Calculate Average Scores

- Function Signature: `void calculateAverageScores(int scores[][MAX_EXAMS], double averages[], int numStudents, int numExams)`
- Description: This function will calculate the average score for each student based on their exam scores and store the results in the `averages` array.
- Input:
  - `scores[][MAX_EXAMS]`: A 2D array containing exam scores for each student.
  - `averages[]`: An array to store the average score for each student.
  - `numStudents`: The number of students.
  - `numExams`: The number of exams per student.

Task 3: Calculate Variance and Standard Deviation

- Function Signature: `double calculateVariance(int scores[][MAX_EXAMS], double averages[], int numStudents, int numExams)`
- Description: This function will calculate the variance of exam scores across all students and exams.
- Input:
  - `scores[][MAX_EXAMS]`: A 2D array containing exam scores for each student.

    - `averages[]`: An array containing the average score for each student.
    - `numStudents`: The number of students.
    - `numExams`: The number of exams per student.
- Output: The calculated variance as a double value.

Task 4: Calculate Standard Deviation

- Function Signature: `double calculateStandardDeviation(double variance)`
- Description: This function will calculate the standard deviation based on the provided variance.
- Input: `variance`: The variance of exam scores.
- Output: The calculated standard deviation as a double value.

Task 5: Display Student Statistics

- Function Signature: `void displayStudentStatistics(string names[], double averages[], double standardDeviation, int numStudents)`
- Description: This function will display student statistics, including their names, average scores, and standard deviation.
- Input:
    - `names[]`: An array containing student names.
    - `averages[]`: An array containing average scores for each student.
    - `standardDeviation`: The standard deviation of exam scores.
    - `numStudents`: The number of students.

Task 6: Find Highest Score

- Function Signature: `int findHighestScore(int scores[][MAX_EXAMS], int numStudents, int numExams)`
- Description: This function will find and return the highest exam score among all students and exams.
- Input:
    - `scores[][MAX_EXAMS]`: A 2D array containing exam scores for each student.
    - `numStudents`: The number of students.
    - `numExams`: The number of exams per student.
- Output: The highest exam score as an int value.

Task 7: Find Lowest Score

- Function Signature: `int findLowestScore(int scores[][MAX_EXAMS], int numStudents, int numExams)`
- Description: This function will find and return the lowest exam score among all students and exams.
- Input:
    - `scores[][MAX_EXAMS]`: A 2D array containing exam scores for each student.
    - `numStudents`: The number of students.

    - `numExams`: The number of exams per student.
- Output: The lowest exam score as an int value.

Task 8: Display Histogram

- Function Signature: `void displayHistogram(int scores[][MAX_EXAMS], int numStudents, int numExams)`
- Function to display a histogram of exam scores for each student.
- Input:
    - `scores[][MAX_EXAMS]`: A 2D array containing exam scores for each student.
    - `numStudents`: The number of students.
    - `numExams`: The number of exams per student.

Task 9: Calculate Grade

- Function Signature: `char calculateGrade(double average)`
- Description: This function will calculate the grade based on the average score for a student using the provided grading scale. It will return the grade as a character.
- Input: `average`: The average score of a student.
- Output: The calculated grade as a character ('A', 'B', 'C', 'D', or 'F').

Task 10: Display Grades

- Function Signature: `void displayGrades(string names[], double averages[], int numStudents)`
- Description: This function will calculate and display grades for all students based on their average scores.
- Input:
    - `names[]`: An array containing student names.
    - `averages[]`: An array containing average scores for each student.
    - `numStudents`: The number of students.

Task 11: Binary Search

- Function Signature: `bool binarySearch(int sortedScores[], int numExams, int target)`
- Description: Implement a binary search algorithm to search for a specific exam score (`target`) in a sorted array of scores.
- Input:
    - `sortedScores[]`: An array containing sorted exam scores.
    - `numExams`: The number of exams.
    - `target`: The exam score to search for.
- Output: Return `true` if the target score is found, or `false` if it is not found.

Main Function:

- Implement the main function to perform the following tasks:
  - Call `inputStudentDetailsAndSort` to input student details and sort exam scores.
  - Call `calculateAverageScores` to calculate average scores.
  - Call `calculateVariance` to calculate variance.
  - Call `calculateStandardDeviation` to calculate standard deviation.
  - Call `displayStudentStatistics` to display student statistics.
  - Call `findHighestScore` to find and display the highest exam score.
  - Call `findLowestScore` to find and display the lowest exam score.
  - Call `displayHistogram` to display a histogram of exam scores.
  - Call `displayGrades` to calculate and display grades for all students.
  - Perform binary search for a specific exam score using `binarySearch`.

Note: You should ensure that the program can handle a maximum of 100 students and 10 exams per student (as defined by `MAX_STUDENTS` and `MAX_EXAMS`).

Feel free to use this assignment template as a guide for your programming project. Students can complete the tasks and gain a comprehensive understanding of sorting, searching, and statistical analysis in C++.

Sample Output:

Enter the number of students: 3

Enter the number of exams per student: 4

Enter the name of student 1: Alice

Enter exam scores for student 1: 85 92 78 88

Enter the name of student 2: Bob

Enter exam scores for student 2: 77 64 90 82

Enter the name of student 3: Carol

Enter exam scores for student 3: 92 88 94 79

Student Statistics:

Student Name Average Score

Alice 85.75

Bob 78.25

Carol 88.25

Standard Deviation: 8.59

Highest Exam Score: 94

Lowest Exam Score: 64

Histogram of Exam Scores for Each Student:

Student 1: [0-19]: 0 [20-39]: 0 [40-59]: 0 [60-79]: 1 [80-99]: 3

Student 2: [0-19]: 0 [20-39]: 0 [40-59]: 0 [60-79]: 2 [80-99]: 2

Student 3: [0-19]: 0 [20-39]: 0 [40-59]: 0 [60-79]: 1 [80-99]: 3

Grades:

Student Name Grade

Alice B

Bob C

Carol B

Enter a score to search for: 85

Score 85 found for Student 1