**Algorithm**

**Start**

Begin the program.

**Starting Interface**

Display a welcome message.

Provide an overview of the project, including its purpose and menu options.

**Variables**

**Global Parameters:**

Declare an integer constant NUM\_TESTS and set its value to 5.

Define an enumeration constant:

Values: Add (1), Remove (2), Display (3), Search (4), Results (5), and

Quit (6).

**Create a structure named Student**

Full name (as a single string).

Student ID (integer, 8 digits).

Number of tests taken (integer).

Dynamic array for test scores (integer pointer).

Average score (float).

**Menu Display**

Create a function to display the menu:

Add a student.

Remove a student.

Display all students.

Search for a student by ID.

Export results.

Quit the program.

Main Program

**Initialize Variables:**

Use a dynamic array for student records.

Declare variables for user choice and flag for program termination.

**Loop:**

Display the menu and prompt the user for a choice.

Use a switch-case structure to handle user input:

**Case 1: Add Student**

Call the add\_Student function.

**Case 2: Remove Student**

Prompt the user for a student ID.

Call the remove\_Student function.

**Case 3: Display Records**

Call the display function.

**Case 4: Search Student**

Prompt for a student ID.

Call the search function.

**Case 5: Export Results**

Call the exportResults function.

**Case 6: Quit**

Display a goodbye message and terminate the program.

**Default:**

Display an error message and loop back to the menu.

**Supporting Functions**

**add\_Student:**

Prompt for and read student details (name, ID, number of tests, and scores).

Dynamically allocate memory for test scores.

Append the data to student.dat.

**remove\_Student:**

Prompt for the student ID.

Use getNumber to determine the number of records.

Create a dynamic array of Student type.

Copy all data except the record with the specified ID back to student.dat.

**display:**

Use getNumber to load all records into a dynamic array.

Print student details in a formatted table.

**search:**

Prompt for a student ID.

Read through the file to find and display matching records.

**exportResults:**

Open averages.dat for writing.

Calculate each student's average after removing the lowest score.

Write student IDs and averages to averages.dat.

**findMinimum:**

Return the smallest test score from an array.

If fewer than 5 tests, return 0 as the minimum.

**End**

Ensure proper file closure and memory deallocation.

**Exit the program.**