Project 3 Algorithm Design

1. Header File (`jjb0363Project3_header.h`) :

   - Contains function prototypes, structure declarations, and constant definitions.

   - Included in all source files (`jjb0363Project3_main.cpp`, `jjb0363Project3_func.cpp`, `getNumber.cpp`).

2. Main Source File (`jjb0363Project3_main.cpp`) :

   - Contains the `main()` function.

   - Includes the header file `jjb0363Project3_header.h`.

   - Calls functions defined in `jjb0363Project3_func.cpp` based on user input.

   - Utilizes `Menu` enum to handle user choices.

3. Function Source File (`jjb0363Project3_func.cpp`) :

   - Contains the implementations of functions declared in `jjb0363Project3_header.h`.

   - Includes the header file `jjb0363Project3_header.h`.

   - Defines functions like `addStudent()`, `removeStudent(int)`, `display()`, `search(int)`, and `exportResults()`.

4. Utility Source File (`getNumber.cpp`) :

   - Contains the implementation of the `getNumber()` function, used to count the number of students in the file.

   - Includes the header file `jjb0363Project3_header.h`.

Code Structure and Algorithm:

1. Initialization :

   - Include necessary header files like `<iostream>`, `<fstream>`, and `<string>`.

   - Define constants such as the number of tests (`NumOfTest`) and the menu options (`Menu` enum).

   - Declare the `Student` struct to hold student information.

   - Declare function prototypes for adding, removing, displaying, searching, and exporting student data.

2. Add Student Function (`addStudent()`) :

- Prompt the user to enter the details of a new student: last name, first name, ID, and the number of tests taken.

- Allocate memory for storing test scores dynamically based on the number of tests taken.

- Write the student's information to the "student.dat" file.

- Close the file after writing.

3. Remove Student Function (`removeStudent(int StudId)`) :

- Read the total number of students from the "student.dat" file using `getNumber()` function.

- Create an array of `Student` structures dynamically.

- Read each student's data from the file and store it in the array.

- If the specified student ID matches any student in the array, remove that student's entry from the file.

- Update the "student.dat" file with the modified data.

- Free allocated memory.

4. Display Function (`display()`) :

- Read the total number of students from the "student.dat" file.

- Create an array of `Student` structures dynamically.

- Read each student's data from the file and store it in the array.

- Display the student information, including name, ID, and test scores.

- Free allocated memory.

5. Search Function (`search(int SearchStudId)`) :

- Open the "student.dat" file for reading.

- Read each student's data from the file until the specified student ID is found.

- If found, display the student's information.

- Close the file and free allocated memory.

6. Export Results Function (`exportResults()`) :

- Read the total number of students from the "student.dat" file.
- Create an array of `Student` structures dynamically.
- Read each student's data from the file and store it in the array.
- Calculate the average score for each student and write it to the "average.dat" file.
- Free allocated memory.

7. Utility Functions :

- `findMinimum(int Scores[], int TestTaken)`: Find the minimum score among the test scores.
- `getNumber()`: Count the number of lines (students) in the "student.dat" file.

8. Main Function :

- Display a menu with options to add, remove, display, search, export results, or quit.
- Execute the corresponding function based on the user's choice.