Project 3 Loop

1. Declare, Initialize globals
    a. const int number of test = 5
    b. const enum menu choices Add=1 ,Remove=2, Display = 3, Search=4, results=5, quit=6
    c. Struct Student { string student, int student id, int number of test taken by the student, int* test scores = new student [number of tests taken by student] (its a pointer) };
2. Main Function:
    a. Loop This until case quit is selected
    b. Display a menu for the user
        i. add student
        ii. remove student
        iii. display student records
        iv. Search for students using ID
        v. Export the average to an output file
        vi. quit the program
    c. ask the user the choice a menu selection
    d. switch case (enum menu choices name)
        i. switch case using enum name not number
            1. MenuSelect my_Select = static_cast<MenuSelect>(choice);
        ii. case add:
            1. call add_student function
        iii. case remove:
            1. ask the user for a student ID to remove
            2. call remove_student function

            a. pass student ID entered by user

        iv. case display:

            1. call display function

        v. case search:

            1. ask the user for the student ID of the student to search for
            2. call search function

                a. pass student ID entered by the user

        vi. case results:

            1. call exportResults function

        vii. case quit:

            1. exit program with a suitable message

        viii. default:

            1. this is the case if the user puts in a menu selection that does not exist
            2. a print message asking for an actual menu selection case

    e. Stop the loop

3. add_Student
    a. with a suitable message asking the user for first name, last name, student ID, Number of tests taken, and test scores of a student
    b. HINT: test scores in dynamic array and size is the number of tests allocates memory accordingly
    c. open the file student.dat and write the data of the student into the student.dat maintaining the original format
    d. use struct members to write the data to the file
    e. will call the main function
4. remove_Student (Parameters: Student ID)

a. call getNumber function
b. create a dynamic array of struct Student. use the number of students (getNumber) for the size of the array
c. open student.dat for reading
d. IN LOOP, read file line one at a time and store data in the appropriate dynamic array
    i. while reading check the student ID for matches to the student ID that the user wants to be removed
    ii. If matched, use bool to store that the program has found a match
    iii. copy the entire file to a dynamic array whether or not a match is found
e. close file
f. if the match is found
    i. open student.dat for writing
    ii. write contents of the dynamic array into the file, only if the student ID doesn't match the student ID being removed
g. if the match is NOT found
    i. print message to indicate that the student doesn't exist
h. close file
i. the function is going to be called to the main

5. display
    a. open data file for resing
    b. using getNumber, the pointer of Sturct STudent, create a dynamic array to read the contents of the file into an array using a loop
    c. in the second loop, display the contents of the array in the following format
        i. 30 spaces for the entire name
        ii. 15 spaces for student ID

        iii. 5 spaces for test scores
        iv. Dont display the number of tests
    d. close file
    e. calling to the main function
6. Search (parameters: student ID)
    a. open data file for reading
    b. declare a pointer of type Student. DO NOT NEED AN ARRAY
    c. in a loop, read each line of the file, and store data in the appropriate struct student pointer member
    d. check if the student ID being read from the file matches the student ID wanting to be searched.
    e. If there is a match
        i. set bool to true and indicate match has been found
        ii. display data of matched student
            1. 30 spaces for the entire name
            2. 15 spaces for student ID
            3. 5 spaces for test score
            4. DO NOT DISPLAY THE NUMBER OF TESTS
    f. if there is not a match (bool is false)
        i. display an appropriate message for the user
    g. the function will be called in the main function
7. exportResults
    a. open averages.dat for writing (this file will contain averages of test scores for each student along with the student ID)
    b. open student.dat for reading

c. create a dynamic array of struct Student using a pointer and call function getNumber to store the contents of the data file student.data
d. in the loop, read the contents of the file, and store them in the dynamic array
e. in the second loop, process each student at a time to compute average
    i. compute the sum of test scores
    ii. note that the minimum score is dropped. call findMinimum to find the minimum score and subtract it from the total score.
    iii. divide the sum by the appropriate integer to compute the average
    iv. write student ID and average to averages.dat
        1. the average column should have one digit after the decimal point
f. close all files
g. the function will be called to the main

8. findMinimum (parameters: array of ints, size of the array) (will compute minimum out of the test scores and returns the score)
    a. if the student has taken fewer than 5 tests the minimum is 0
    b. if the student has taken all 5 tests, the minimum score is the minimum of the 5 scores in the array
        i. make sure to subtract 1 from the total number of tests taken for the calculation of averages
    c. will be called to export function
9. create file euidProject2_main.cpp - copy main function to file
10. creat file euidProject2_func.cpp - copy all functions (other than main function) to this file
11. create file euidProject2_header.h - copy all globals, include fields and function headers to the header file make sure to include the header file in the .cpp file
12. submit all 3 files
13. compile the files together

14. Deallocate all dynamic memory programming throughout the code
15. ALL THE FUNCTIONS
    a. getNumber
    b. findMinimum
    c. add_Student
    d. remove_Student
    e. display
    f. Search
    g. exportResults
    h. main