Lab04

1. Lab Topics

This lab covers different topics, but more emphasis is placed on good use of relational operators, logical operators, decision branching (nested and chained if-else statement), loops (while, do while, for and infinity loops).

2. Grading System

Write a C++ program to simulate a simple grading system for a class of students. The program should take the number of students as input and then repeatedly prompt the user to enter each student's score. After entering all scores, the program should calculate and display the following statistics:

1. The average score for the class.
2. The highest score and the student's name who achieved it.
3. The lowest score and the student's name who scored it.
4. The number of students who passed (scored 60 or higher).
5. The number of students who failed (scored below 60).

Requirements:

1. Use a *do-while* loop to prompt the user for the number of students until a valid positive integer (1 – 100) is entered. Similar to input validation taught in class.
2. Use a *for* loop to iterate through each student's scores.
    a. Check to make sure the score entered is also between 0 and 100 using the appropriate relational and logical operators.
    b. If the wrong score is entered, remember to increment/decrement the count such that the next value that will be entered will be for the same student and not the next.
    c. Subsequently collect the name of students.
    d. While collecting the user's input, you could keep track of the highest and lowest score and continually check if any new value entered is now the highest or lowest value.
    e. You should also keep track of the student's name with the highest and lower score, and it can be overridden once there is a new score that is the highest or lowest. Appropriate use of decision branching is necessary here.
    f. Keep track of the number of scores that are above and below 60. This can be done with decision branching and increment operators.
3. Use an infinite loop to continuously run the program until the user chooses to exit.
    a. Use a do while loop.

b. Ask if the user want to continue running the program. If the user's input is Yes, YES, yes, Y, **or** y, continue running the program, else end the program "gracefully".

General Hints

1. Remember to use a smaller value for the number of students so that you could easily test your code.
2. Use appropriate arithmetic operators to determine the average score, relational operators and logical operators to determine pass/fail and calculate statistics.
3. Implement decision structures with if-else statements, including nested if-else statements, if necessary, to track the highest and lowest scores and the names of students who passed and failed.
4. For this program, we are assuming that the user will only enter the **type** of input required. For example, the user won't enter a string when score is being asked for.
5. **DO NOT** use arrays, structs, classes or any other concept not yet thought in class.

BONUS (10 points):
Capture the error that can arise in number 4 of the general hints. Use exception handling. You can read about it online and learn how to use it.
Note: No help will be provided for this bonus. Students are expected to research about it and learn how it is being used.

Example Output 1 (Testing the program):

```
Enter the number of students (1 - 100): 0
Invalid input. Please enter a number between 1 and 100.
Enter the number of students (1 - 100): 5000
Invalid input. Please enter a number between 1 and 100.
Enter the number of students (1 - 100): 4
Enter the name of student 1: Yu
Enter the score for Yu: 1000
Invalid score. Please enter a score between 0 and 100.
Enter the score for Yu: -90
Invalid score. Please enter a score between 0 and 100.
Enter the score for Yu: 400
Invalid score. Please enter a score between 0 and 100.
Enter the score for Yu: 40
Enter the name of student 2: Mi
Enter the score for Mi: 70
Enter the name of student 3: Yumi
Enter the score for Yumi: 60
Enter the name of student 4: Miyu
Enter the score for Miyu: 90
```

```
Class Statistics:
Average Score: 65
Highest Score: 90 (Student: Miyu)
Lowest Score: 40 (Student: Yu)
Number of Students Passed: 3
Number of Students Failed: 1

Do you want to continue (Yes/No), (Y/N)? yEs
Exiting the program gracefully.
```

Example Output 2 (Testing the program):

```
Enter the number of students (1 - 100): 50
Enter the name of student 1: Max
Enter the score for Max: 78
Enter the name of student 2: Maxine
Enter the score for Maxine: 45
Enter the name of student 3: Jason
Enter the score for Jason: 45
Enter the name of student 4: Jasmine
Enter the score for Jasmine: 78
Enter the name of student 5: . . .

You really don't expect me to continue for 50 students. ☺
```

Example Output 3:

```
Enter the number of students (1 - 100): 3
Enter the name of student 1: Mi
Enter the score for Mi: 67
Enter the name of student 2: Yu
Enter the score for Yu: 46
Enter the name of student 3: Miyu
Enter the score for Miyu: 89

Class Statistics:
Average Score: 67.3333
Highest Score: 89 (Student: Miyu)
Lowest Score: 46 (Student: Yu)
Number of Students Passed: 2
Number of Students Failed: 1
Do you want to continue (Yes/No), (Y/N)? Yes

Enter the number of students (1 - 100): 4
Enter the name of student 1: Mi
Enter the score for Mi: 12
Enter the name of student 2: Yu
Enter the score for Yu: 67
Enter the name of student 3: Yumi
Enter the score for Yumi: 95
Enter the name of student 4: Miyu
```

```
Enter the score for Miyu: 58

Class Statistics:
Average Score: 58
Highest Score: 95 (Student: Yumi)
Lowest Score: 12 (Student: Mi)
Number of Students Passed: 2
Number of Students Failed: 2
Do you want to continue (Yes/No), (Y/N)? n
Exiting the program gracefully.
```

3. How to get full marks

To get a 100% on this lab your code should:

1. Use good variable names such that one can easily understand a variable's purpose just by looking at the name.
2. The program needs to be intuitive (e.g., display proper messages while you are taking user input or printing the result)
3. Follow all good coding conventions such as proper indentation.
4. Adhere to all coding standards outlined in lab2.
5. Follow the instructions of cloning, making dir, and submitting your code to git as previously discussed in lab01 and lab02
6. Comment your code properly (do not write comments for things that are obvious)
7. Push your most recent code in git and submit through canvas as well. The canvas submission should include following files:
    a. The cpp file downloaded from git
    b. **At least 2 Image files**
        i. **Screenshot of the right result**
        ii. **Screenshot of testing "abnormal" values for the user's input.**