CS/CE 1337 – PROJECT 2 – Cash Cab Contestant Search

Pseudocode Due: 9/26 by 11:59 PM

Core Implementation Due: 10/2 by 11:59 PM

Project Due: 10/9 by 11:59 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission and Grading:

- The pseudocode will be submitted in eLearning as a Word or PDF document and is not accepted late.
- All project source code will be submitted in zyLabs.
    - Projects submitted after the due date are subject to the late penalties described in the syllabus.
- Programs must compile using gcc 7.3.0 or higher with the following flags enabled
    - -Wall
    - -Wextra
    - -Wuninitialized
    - -pedantic-errors
    - -Wconversion
- Each submitted program will be graded with the rubric provided in eLearning as well as a set of test cases. These test cases will be posted in eLearning after the due date.
    - zyLabs will provide you with an opportunity to see how well your project works against the test cases. Although you cannot see the actual test cases, a description will be provided for each test case that should help you understand where your program fails.
- **Type your name and netID in the comments at the top of all files submitted. (- 5 points)**

Objective:

- Allocate memory dynamically.
- Utilize pointers to directly manipulate memory.
- Implement smart pointers

Problem: With the resurgence of game shows, a decision has been made to reboot *Cash Cab*. In order to find contestants, the producers are launching an online search where potential contestants complete a multiple-choice test to be considered for the show. As part of the company creating the test, you have been assigned the task of generating a report for each participant and calculating statistics for each version of the exam.

Pseudocode: Your pseudocode should describe the following items

- Main.cpp
    - Detail the step-by-step logic of the main function
    - List other functions you plan to create
        - Determine the parameters
        - Determine the return type
        - Detail the step-by-step logic that the function will perform

- A list of at least 10 test cases you will check during testing
    - Specific input is not necessary
    - Describe what you are testing

Zybooks Information:

- Your source file will be named `main.cpp`
- Core implementation has unlimited submissions
    - This will help you make sure the basic actions of your program are working properly in the Zybooks environment
- Final submission is limited to 12 submissions

Core Implementation

- Read answers from file
- Read contestant data from file
- Calculate contestant score
- Display contestant data (ID, score, incorrect contestant answers, correct answers)
- Not required
    - Smart pointers
    - Mean, median and mode
    - Most questions missed

Details:

- Input will consist of 2 files
    - A file for correct answers
    - A file for contestant responses
    - Start the program by prompting the user for the answer key filename and then the contestant responses filename
- All arrays must be dynamically allocated (-10 points if not)
- All array access must be done with pointer notation using pointer arithmetic (no brackets or offset notation) (-10 points if bracket notation used)
- EXTRA CREDIT (+10 points): Use smart pointers for all pointer interaction.
    - Exception to the requirement above – offset notation is allowed because pointer arithmetic cannot be used on smart pointers
- For each contestant
    - Compare the contestant's answers to the correct answers
    - Create a report for the following information:
        - Contestant's ID number
        - Score
            - Score = # of correct answers / total # of questions * 100
        - List of questions missed, including the correct answer and contestant answer
- After all contestant responses have been reviewed, create a statistical report
    - Calculate the overall mean, median and mode of all contestants' scores
        - Mean = average

        - Median – a value in which half of the data set is less than the value and half of the of the data set is larger than the value
            - If the size of the data set is even, the median is halfway between the two middle values
            - If the size of the data set is odd, the median is the middle value
            - Calculating the median will require sorting the array
        - Mode – the value that has the highest number of occurrences in the data set
            - A sorted data set will help here
            - Multiples of the same value will be placed together forming a group.
            - Look for the group(s) with the largest size
    - Identify all questions that were missed by more than 60% (inclusive) of the contestants
- All calculated values will be displayed to 2 decimal places

Input:

- Answer key
    - The file consists of a series of characters separated by white space (not necessarily a newline).
    - Each non-whitespace character represents the correct answer for a multiple choice question.
    - The first non-whitespace character is the answer for the first question, the second non-whitespace character for the second question and so forth.
    - Each line in the file will end with a newline (except the last line which may or may not have a newline)
- Contestant responses file
    - Each line of the file will contain data for a unique contestant.
    - Each line of the file will begin with a contestant ID (a 10-digit number – can be treated as a string) followed by the contestant's answers for the test.
        - Each answer will be separated by a space and there will be exactly as many answers as there are questions listed in the answer file.
    - Each line in the file will end with a newline (except the last line which may or may not have a newline)
- No input validation will be required

Output:

- All output will be written to the console.
- For each contestant, the following information will be written (each bullet represents a separate line):
    - `<Contestant ID><space><hyphen><space><score>`
    - If the contestant answered any questions incorrectly
        - List the number of each incorrect question separated by spaces
        - List the incorrect answers given by contestant
            - Line up each answer with the ones digit of the number on the previous line
        - List the correct answers
            - Line up each answer with the incorrect answer on the previous line
    - 1 blank line
- Create a summary at the end of the report

- `Mean:<space><mean>`
- `Median:<space><median>`
- `Mode:<space><mode>`
    - If there is more than 1 value for mode, list all modes in ascending order separated by a comma and space
- Blank line
- Most missed questions header
    - `MOST MISSED QUESTIONS`
- For each question that has a 60% or higher miss rate
    - `<question number><tab><percentage><percent sign><newline>`
        - Percentage = percentage of contestants that missed the question