CS2308 - Foundations of Computer Science II
Program Assignment #1
Total Points: 100

Due 1 second before class begins, Feb. 20, 2020

In this project, we will write a program that controls a quiz show, much like the many popular TV shows. The program will read in a group of questions and their multiple choice answers, storing them to a 2D array of strings (which can be implemented as a 3D array of type char or a 2D array of C++ strings.) The correct answers will be stored in a separate 1D array of type char.

The rules of play.

- Randomly present a question to the player, followed by its four possible answers (identified as A-D).
- Player selects an answer (program validates it's in range A-D, otherwise prompts for answer again.)
- If correct choice, award points (see below).
- If incorrect, offer player another chance. If player elects to try again, display the question, omitting the previous incorrect choice.
  - If player gets second attempt correct, award ½ the points that would be applicable.
  - If player gets second attempt wrong, game is over, all points forfeited!
  - If player chooses to skip the missed question, go on to next random question, no penalty.
- The maximum number of questions per player round is 6, which will potentially win over a million points.

Scoring

- Points awarded for each question go up by a multiple of 10 of the points awarded on the previous question, starting out at 10 for the first one.
- If a player took a second chance at a question and got it right, the points awarded will be ½ of 10 times the previous award, and future points will build from that amount.
- If a player skips a question he missed the first attempt on, points to be awarded on next question remain unchanged (10 time previous points award).
- Example sequences of points awarded:
  - 10 100 1000 10000 100000 1000000 perfect scored = 1,111,110
  - 10 100 500 5000 50000 500000 missed one, then perfect = 555,610
  - 10 100 0 1000 10000 100000 skipped one = 111,110

The Data Files

Two files will be used, one containing the questions and responses, the second containing the letters of the correct responses (the answers.) The Question file will be a series of text strings in the form:

This is a question that the player must answer.
This is the first possible choice.
This is the second possible choice

Guess what, this is the third possible choice
Finally, the fourth choice
(one or more blank lines separate each question group)
(…and more sets of questions follow)

There will be 0 to 50 sets of questions in this format in any data file. Accompanying the question bank will be a file with a series of characters, one for each of the five-line question sets (thus, 0 to 50 characters in the file.) These characters, in the range A-D, will indicate the correct answer for the questions.

For this assignment, you are guaranteed that in the question bank, there will be no incomplete sets of questions. The answer file may, however, not agree with the question file, that is, it may have more or less than the number of questions' worth of characters. In that event, display a meaningful error message and end the program.

To handle the blank line(s) between question sets, and any blanks that may exist at the end of the file, you should read a line. If the line has a non-zero length, you have found a question. It will be immediately followed by four more valid lines (the responses), which you can assume will be easily handled by a for loop.

The names of the question file and the answer file will be given on the command line, as well as an integer seed for srand. Your program must check that sufficient arguments are present before attempting to open the files. If the argument list is too few or too many, display a meaningful error message and exit. If either of the input files fails to open, display a meaningful error message and exit. If the question file is empty, display an error message and exit.

After the player has answered all questions, store the current player's name and score to a file named "summary.txt". Use the append mode so that every player's score is added to the file – that is, it will become a history of all who've played the game. At the end of the program, read in the names and scores of all players from "summary.txt" (the number of players who have played the game will not be known until reading to the end of the file), sort the scores and names accordingly, display the name and score of the player with the highest score, and the rank (1st place, 2nd place, 3rd place, etc) of the current player.

Data Storage
You will store the questions and responses in a 2D array of strings. No string will be longer than 150 characters. You may use C-style strings (character array based, NULL terminated) which implies you will need a 3D array. You may also use C++ style strings, which will be stored as a 2D array of that type. The answers will be stored in a 1D array of characters.

Displaying a question
When presenting a question to the player, format it as shown below. You will display the letters A-D in front of the responses; they are not included in the response text. Your program will randomly select questions from those read in, keeping track of which questions have been used. Do not repeat a question during the course of the game play. When a player is making a second attempt at a question, leave the line of the incorrect selection blank, but keep the same letters

assigned to the responses as in the first attempt. You must present the answer choices in the order in which they are read in, do not mix them up.

Random Number Generation
Before generating a random number use argv[3] to seed srand() with an integer. Seeding the random number generator with a fixed number allows you to test the other parts of your code while fixing the order of numbers that will be generated.

Example of a question's first display:

```
Joe Student    Here's Question Number 1
Where does a C++ program begin is execution?
A.  The first function in the file
B.  The first function by alphabetic order of names
C.  The function main( )
D.  The function specified in the #DEFINE FirstFunction

Your choice? > D
```

Example of a question being redisplayed

```
Joe Student    Here's Question Number 1 (second try)
Where does a C++ program begin is execution?
A.  The first function in the file
B.  The first function by alphabetic order of names
C.  The function main( )

Your choice? >
```

Required Functions
Your program must have at least the following functions. You may create additional ones to further break the program down in to manageable units.

Main( )
Your main( ) function should be just a high level organizer. It should check the command line arguments, call the functions to read the input files, check that the questions and answer files have the same number of items, call the play_game( ) function, write the score to the summary file and display the final score message (highest score and the current player's rank) to the player. No other significant activity should take place in main( );

read_questions ( )	reads in the questions/responses from file, storing to 2D array of strings
read_answers( )	reads in the answers from file, storing to a 1D array of characters
show_question( )	displays a question and its responses, labeled A-D, optionally hiding a specified response for the second try at a question. Its primary parameter is an array of strings that is one single question/response set

player_try( )	handles the overall display of a question, getting the response from player, validate response is in range A-D
play_game( )	the core function for controlling the game play. Picks questions, checks correctness of player response, offers second chance, determines and applies scoring. (NOTE: This function does a lot – it will not be unusual to see this on at twice the recommended size for a function.)
sort_score( )	read in historical records of previous players (including current player as well). Sort the scores and names accordingly, display the player's name and score who has the highest score, and the rank of the currently player based on his/her score.

~~~~~~

Timeline:

Jan. 30:	main() , read_questions( ) & read_answers( ) set up. Stubs for other major functions done.
Feb. 6:	show_question( ) & player_try( ) implemented.
Feb. 13:	most of play_game( ) done, player_try( ) and sort_score() completed.
Feb. 18:	All functions work. Full test your program on the Linux server.
Feb. 20:	Due Date. It should be fully commented and follow the coding standards.

Comments and Suggestions:

DO NOT DELAY. Start writing the program from Day 1. If you wait until the night before the due date, you will have a miserable night and it is less likely you could complete the project.

Do not try and write the entire program all at one time. Work on the program in small sections, as the timeline indicates.

Notes:

- **Your program should accept both upper case and lower case user choice. Make sure to use toupper() or tolower() function to simply your code.**
- Your program must correctly compile and is executable.
- Be sure your code file is readable and neat. Do not allow lines to extend past 80 characters, use appropriate white space and make sure to use a consistent indentation scheme shown in coding standard.

Program Submission

Please submit only your source file (firstname_lastname_prog1.cpp) to TRACS. DO NOT submit the entire project! You will get zero if you fail to submit your cpp file electronically to the web server before the deadline.

You also need to submit a paper copy of your source code in class at the deadline day as well. If you fail to submit the paper copy in time or your paper copy does not match your electronic copy, you will lose 20% of your total grade every day until the instructor received the correct version of your paper copy.
~~~~~~

Your program will be graded as follows, please make sure to check each item before you submit.

CS2308 Program #1 Name:___________________________________

Program and Run Time Output: ________ (86 Points)

____________ (5) Command Line Processing

- ________ (3) Tests argc - must be 3 – error message and exit on failure
- ________ (1) Uses argv[1] and argv[2] as input file names
- ________ (1) Uses argv[3] as seed

____________ (15) Read Input Files

- ________ (4) Open and check files, display message and exit upon error
- ________ (5) Read questions into a 2D/3D array of strings
- ________ (4) Read answers into a 1D array of char
- ________ (2) Return numbers of records read

____________ (8) Game Setup

- ________ (3) Compares # questions to # answers, message and exit upon error
- ________ (2) Gets player name
- ________ (3) Generate random numbers

____________ (6) Game Control

- ________ (3) Randomly picks questions from available pool with seed for srand
- ________ (3) Does not duplicate question use

____________ (16) Display of Questions

- ________ (3) Show player name followed by question and its number
- ________ (3) Prepend letters A-D before answers (do not add them to answer strings in stored array)
- ________ (4) Hide incorrect answer from previous attempt
- ________ (3) Get player response, validate it's in range A-D
- ________ (3) make user choice non case sensitive (use toupper or tolower function)

____________ (12) Scoring

- ________ (3) Compare player response to corresponding answer
- ________ (3) Offer player second chance on incorrect first response
- ________ (3) Apply correct score to correct response (10* or 5* previous awarded points)
- ________ (3) End game with 0 points if second attempt incorrect

_____________ (3) Write Summary File

________ (2) Open file in append mode and correct file error handling.

________ (1) Append player name and score to the end of the file

_____________ (12) Sort and Display Final Scores

________ (3) Open the summary file, read all previous records and return the total number of records

________ (7) sort all players record () – either bubble sort or selection sort

________ (1) display the highest record information (name and score)

________ (1) display the current player's rank (1st , 2nd, 3rd , etc)

________ (4) play_game( ) primary control of game play – selects random ques., scoring, 2nd chance

________ (3) main ( ) correct control of the entire program and display meaningful messages when errors occur

________ (2) correct function prototype

Coding Standards: ________ (14 Points)

_____________ (3) Documentation (program and function headers)

_____________ (3) Comments

_____________ (3) Meaningful Variable and Function Names

_____________ (2) Indentation Scheme / Use of { }

_____________ (2) Modularity (organization of program segments, use of functions)

_____________ (1) Print this grading rubric and staple it with the paper copy submission.

Total: ________ (100 Points)

Executable Version: ________ (%)

If your code cannot be compiled or executed, your final score will only be 50% of your accumulated score of each item above.

Final Score: ________ (100 Points)