

CPTR150 Computer Science 1

Final Assignment

When you procrastinate to the last possible day on an assignment



Due Date: Sunday 16 April.

Objective:

This assignment provides the opportunity for you to demonstrate your understanding of the programming concepts explored throughout the semester, and to enhance your ability to be creative and resourceful.

This final assignment is worth 40% of your overall grade.

Problem specification:

You will be developing an application that simulates a debate. Following the same process as your own debate for your research assignment. Using an opposing turn, affirmative turn and a question turn (judges).

Program Flow

- The program starts with a welcome message and gives the user the option to load the previous debate result, start a new debate or end the program.
- If the user chooses to load the previous debate result, load from the file the results of a previously generated debate.
- If the user chooses to start a new debate. Display the debate question and allow the user to enter a start key, any other entry will end the program.
- Debate process:
 - Presentation of arguments. This will be done in 3 turns.
 - The first argument from the affirmative team will display.
 - The rebuttal from the opposing team will display.
 - The judges turn. You need a marker to identify each:
 - The first question is displayed for the affirmative team, and the teams' response is displayed.
 - The first question is displayed for the opposing team, and the teams' response is displayed.
 - The second question is displayed for the affirmative team, and the teams' response is displayed.
 - The second question is displayed for the opposing team, and the teams' response is displayed.
- Scoring and result
 - At the end of the debate process, the scoring should be displayed as follows (sample output provided below):

- The score for each team for each turn is displayed.
- The total score for each team is displayed.
- The winner of the debate is displayed.

Round One:	Affirmative: 5	Opposing: 3
Round Two:	Affirmative: 5	Opposing: 3
Round Three:	Affirmative: 5	Opposing: 3
Question Round One:	Affirmative: 5	Opposing: 3
Question Round Two:	Affirmative: 5	Opposing: 3
TOTAL SCORE:		
	Affirmative: 20	
	Opposing: 40	
WINNER: Opposing!		

- The results should be written to file – overriding the previous file is acceptable.

Program Requirements

- You need to have three files included in your project:
 - Judges
 - File must contain 4 questions. Two questions for the affirmative team, and two questions for the opposing team.
 - Affirmative
 - File must contain three arguments ‘for’.
 - Two answers that would answer two of the questions in the judges’ file.
 - Opposing
 - File must contain three arguments ‘against’.
 - Two answers that would answer two of the questions in the judges’ file.
- When you run your program, the contents of each file should be loaded into separate arrays. That is, one array for arguments for, one array for arguments against, and one for

the judges' questions. You must check for and display the appropriate message for the following scenarios:

- If the program attempts to load a file that is missing
- If the program fails to load all the required arguments into the array. Note that *arguments for* should load 5, *arguments against* should load 5, and *judges* should load 4.
- The program would start with the argument for, followed by the opposing teams' rebuttal. A random score between 1 and 5 must be generated and stored for each team in each turn.
- The results must be written to a file.

Other Features

- Detailed and appropriate comments explaining your code must be included.
- The program must use a struct. How you decide to use this, is up to you. However, it must be logical and make sense.
- The program must use functions. How you decide to use this, is up to you. However, it must be logical and make sense.
- The program must use references or pointers. How you decide to use this, is up to you. However, it must be logical and make sense.
- Variables, Arrays, Structs and Functions must be appropriately named.

Documentation Requirements

You must write an official report that accompanies your program. The report will contain the following elements:

1. USC official cover page.
2. Introduction. This section contains a brief overview of your application.
3. Algorithm. Plain English, NOT CODE:
4. IPO Chart
5. Errors.

- a. Screenshot **at least 4 errors** that you encountered when doing this assignment. Include the screenshot in your report, explain why the error occurred, and explain what you did to you resolve the error.

Submission Requirements

You are to submit your source code (.cpp file), your text files (affirmative, opposing, judging and results), and a pdf of your final documentation within a Google Drive Folder named 'Final Assignment'. Please ensure that the folder is changed to allow editing rights. Copy and paste the link in the submission space in E-learn.