PROGRAM 1 ASSIGNMENT INSTRUCTIONS

CSC 1310-001, SPRING 2025

STEAM®

DUE DATE

Thursday, February 6, 2025 by 11:59pm

You may submit up to 2 days late with 10 points off per day late.

IMPORTANT RULES YOU MUST FOLLOW SO YOU DO NOT GET A ZERO

As reported in the syllabus, you may not use the following in your code:

- while(true) or while(1)
- Goto statements
- Recursively calling the main function
- exit()
- ternary operators (any syntax requiring a '?')

GENERATIVE AI POLICY FOR PROGRAM 1 ASSIGNMENT

- You may use generative AI **only to help you with troubleshooting the code**. **However, if you do, you must share your entire generative AI conversation**. Either take screenshots of your entire chat conversations, paste to a word document, and include with your submission, or if you have ChatGPT 40, you can Share a public link to your chat by clicking "Share" in the top right, creating the link, and then creating a document where you paste all your chat links and make sure to upload that with your submission.
- **You may NOT copy and paste any part of the programming assignment instructions or sample output**, but you may upload the code to aid in troubleshooting.
- **You may NOT ask generative AI to create functions or any code for you.** This will be considered academic misconduct for this assignment.
- **You may not show more then 1 to 2 lines of code to another person** (parent, other students – even if they are not in our class) who is not a TA or instructor. This will be considered academic misconduct for this assignment.
- If you get ideas on how to write a portion of your code, you must CITE YOUR SOURCE in a comment in that section of your code. You still must be able to explain what the code is doing.

DESCRIPTION

This program serves as a **Steam Game Management System**. It allows users to manage a database of games with the following features:

1. **Display All Games**: Lists all games in the database with detailed information, including name, sales, revenue, publishers, developers, genres, and Steam-specific details like price, review scores, and average playtime.
2. **Filter Games by Genre**: Allows users to view games filtered by their genre, providing an organized way to explore the database.
3. **Add a New Game**: Lets users input detailed information to add a new game to the database, including genres and Steam-specific details.
4. **Delete a Game**: Enables users to select and remove a game from the database.
5. **Save Changes**: Updates the database file (games-list.txt) with any modifications made during the program session.

This program demonstrates concepts of **file handling**, **object-oriented programming**, **dynamic data structures**, and **user input validation**. It is an excellent assignment for practicing **C++ programming skills**, especially in managing complex data relationships and enhancing code organization.

GIVEN FILES

You are given the entire program and required text files.

HEADER FILES GIVEN

- **Game.h** – contains the Game class
- **Genre.h** – contains the Genre class
- **Steam.h** – contains the Steam class

SOURCE FILES GIVEN

- **steamGames.cpp** – contains the whole program, including main function

TEXT FILES GIVEN

- **game-genres.txt** – contains different genres and their descriptions.
- **games-list.txt** – contains game data

DIRECTIONS

STEP ONE:

Fix the code so that it can run with the provided text files exactly like the given sample output (in a separate document called **Program 1 Sample Output.pdf**. You may have to add a few functions to one of the classes or add a little code to one of the functions in the steamGames.cpp source file.

STEP TWO:

Add code to validate user input in the addGame function for all numerical inputs (long long, double, and bool). Refer to the validation while loop in the main function for the menu choice that was already provided for you to help you with this task. Test your code to ensure your input validation works. Test it with strings and numbers. All numerical inputs may be 0, but shouldn't be negative and there is no maximum value.

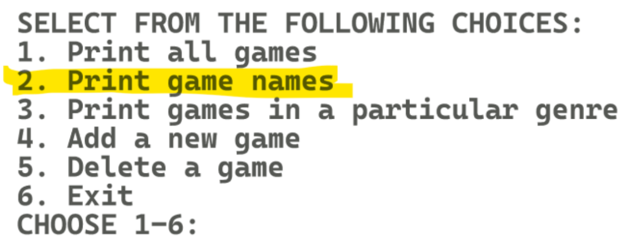
STEP THREE:

Extend the program to allow user to print only the game names instead of all the game details. Make sure that each game is numbered such as:

1. Hogwarts Legacy
2. The Lord of the Rings: Return from Moria
3. Palworld

   ...etc.

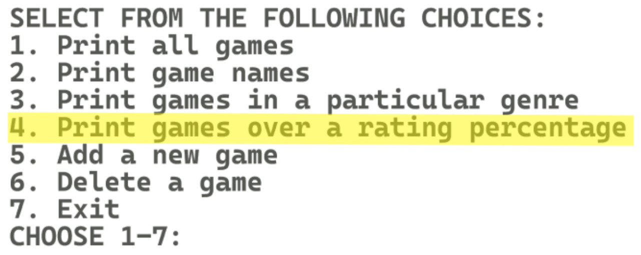
The new menu should show 6 menu options instead of 5 and printing the names should be menu option #2. See the screenshot below for clarification:



```
SELECT FROM THE FOLLOWING CHOICES:
1. Print all games
2. Print game names
3. Print games in a particular genre
4. Add a new game
5. Delete a game
6. Exit
CHOOSE 1-6:
```

STEP FOUR:

Extend the program to allow the user to choose from the menu to print games with a particular rating. The user should be able to select a particular rating percentage (such as 75%), and your program should print all the games (should include the name of the game and the game's review score) that have a rating above 75%. The main menu should be modified again so that printing over a certain percentage should be menu option #4. See the screenshot below for clarification:



```
SELECT FROM THE FOLLOWING CHOICES:
1. Print all games
2. Print game names
3. Print games in a particular genre
4. Print games over a rating percentage
5. Add a new game
6. Delete a game
7. Exit
CHOOSE 1-7:
```

Make sure to validate the input that the user gives you. The user should select a percentage between 0 and 100%.

STEP FIVE:

Extend the program to disallow duplicate game entries (by checking the game name only). In order to do this part of the program, you will have to make modifications to both the readGamesFromFile function and the addGame function to ensure duplicates aren't added in either function.

Please see the "**After All Steps Program 1 Sample Output.pdf**" to see what the sample output should look like after completing all five steps.