CS 1337 – Homework 1 - Snake Game Extensions

Dr. Doug DeGroot's Class

Due: Sunday, February 7, 2020, by midnight

How to submit: Upload your homework to the eLearning site (the .cpp file). Don't zip the program. The project must compile on either Code::Blocks or MS Visual Studio 2015 (or later).

Note: You must work alone on this homework; do not collaborate with other people.

Online Help: I'll try to provide online help through office hours using MS Teams several times to answer any questions and address any problems you might experience.

Objective: Modify and extend the original SnakeGame program in the following ways:

1. Continue to improve the code wherever you think it needs to be improved for clarity or comprehensibility. This is essentially a continuation of Homework 0. Show me what you know about good programming structure and style. (You have probably already done most of this work in HW0.)
2. Check for possible bugs: Is there a bug in the board drawing code that sometimes makes the snake disappear? If there is, find it and fix it. There may be other bugs in either the Logic or Draw routines; find any and fix them.
3. The Logic routine is too complicated for easy understandability. For example, it addresses multiple, separate concerns. Fix this by *separation of concerns*.
4. The snake growing part of the Logic routine is poorly written – unclear variable names, no commentary, and obscure logic. Fix these issues to the best of your ability.
5. Add the capability of having more than one fruit on the board. Create some variable and set it to a value of 2 to 5 or more, say, to represent the number of fruits that should be on the board. Don't hardwire the number of fruits; instead, ask the user at the beginning of the program how many fruits to create. Always create a new fruit for each fruit that is eaten, of course, and at random locations.
6. If the user fails by running into the snake's tail, don't just quit, but instead report this fact to the user and then and only then quit the current game.
7. Add an option to let the user play another game when s/he loses rather than just exiting the program when the user fails.
8. Add an input option P that pauses the game and waits for the user to hit any other key before proceeding.
9. When the program first starts, print out some instructions for using the game to the user.
10. Add an option to the game that makes the player lose the game if the snake hits a wall.
11. Make the input routine case-insensitive.
12. Add any other features you think are valuable/interesting, and note in the comments at the top of the program what you added and why.
13. Use good debugging techniques but leave the debugging code in the program when you submit it; just comment out the debugging code or, preferably, use Boolean switches to turn them off. Make your program talk to you. This is important.

Annotations for the code:

1. The main function can be at either the beginning or the end of the program. I don't care which.
2. Add comments at the top of your main.cpp file to include your name, the name of the program, a changelog, and notes on what changes and improvements you made to the program. (For example, list what bugs you fixed, what new features you added, what code you improved, ideas for future improvement, and so on.)

3. Comment your code effectively, as we discussed in class. Use descriptive variable names everywhere so that the code becomes as self-documenting as possible. But do use commentary to improve readability and comprehensibility.
4. You absolutely MUST use consistent indentation and coding styles throughout the program. Failure to do so will result in a loss of 5 points. And just bite the bullet and use Google-style formatting! 😊
5. If the program does not compile without error or does not work at all, or works but works incorrectly, at least 10 points will be deducted.
6. No late submissions will be accepted – period (except for my one-hour tolerance if absolutely necessary). Please meet the deadline.