Homework 2: Coding a mathematical game

Your assignment is to write a program that allows the user to play a certain game, described below.

The Game

The game has two players, a firefighter and a pyromaniac. They walk together, in a consistent direction, around a circular path lined by n trees. At the beginning of the game, some of the trees are on fire and some are not. Each time the players walk past a tree, one of the players takes a turn, based on the status of that tree:

- If the tree is on fire when they arrive, the firefighter can either extinguish the fire or not.
- If the tree is not on fire when they arrive, the pyromaniac can either light it on fire or not.

The firefighter wins if there is ever a time when all of the trees are extinguished at the same time.[1]

In the classical version of the game, the pyromaniac has no way to win other than the firefighter giving up. In particular, it must **not** be a win for the pyromaniac if all of the trees are on fire simultaneously. (In fact, that will soon be a win for the firefighter, as he will take the next n turns and extinguish all the fires without the pyromaniac getting a turn.) If you want, you can create your own win condition for the pyromaniac. For example, maybe the pyromaniac can win by threefold repetition, similar to the draw rule for chess. (In this game, if the same position is repeated three times, then the firefighter clearly isn't making progress, so arguably that should mean the pyromaniac wins.) Maybe there's a move limit, and the pyromaniac automatically wins after a certain number of moves.

Some possible features

Other than the basic definition of the game described above, most of the choices for what you create are left up to you. You can make it a website, mobile app, or desktop app. You can design the user interface as you see fit. As mentioned in the footnote, you can change the representation of the two states from fires/trees to heads/tails, or something else entirely; you can even code multiple options, and give users the option to choose their favorite. You can make it a two-player game and/or create a one-player mode where the computer controls the other player, or give the player the option to choose either. If you have a computer player, you can decide how difficult to make it, or even include multiple difficulty levels. You can decide whether the pyromaniac has a win condition. You might allow users to track their stats. You can decide how the game is set up: how many trees are there, and which trees are on fire at the beginning? Do you always start from the same starting position (or a few pre-determined

[1] Instead of setting/extinguishing fires, the classic description of this game involves flipping coins to heads/tails. The version in terms of trees and fires is taken from Problem 15 of the Carnegie Mellon University Math Club Problem of the Day: `https://cims.nyu.edu/~tjl8195/CMUMC_POTD_Book.pdf`. If you decide to use this formulation, you should give credit accordingly, though you may also decide to represent the fire/tree states as heads/tails, or any other analogy that you find appealing.

choices), or do you randomly generate a new starting position every time? If you randomly generate starting positions, do you reject trivial starting positions where every tree or almost every tree is on fire? Do you let the user pick a starting position? Do you let the user pick how many trees are involved in the game?

The fact that you have so many choices is an important component of the assignment. You will almost certainly not have time to implement all of the features described above. You will have to decide which features you think are most valuable, and manage your time accordingly. You will probably also want to work in a somewhat agile style, getting the main functionality working before you start adding other features/improvements. I suggest you do some planning in advance to keep your code well-organized, so that it's easy to add additional features after you finish the basics.

What you must submit

You should create and share with me a GitHub repository including all the files necessary to run your program, as well as very clear instructions on how I can play your game to test it.

You should include proper credit where appropriate. For example, you should provide credit for any art assets you might (legally) include, and (as mentioned in a footnote above) give credit to the CMUMC problem book if you use the tree/fire version of the game.

Evaluation

You will be evaluated primarily on how enjoyable your game is to play. For example, the user interface should be pleasant and easy-to-use. This also means that in general, adding additional (worthwhile) features will improve your grade. However, a game with lots of poorly-implemented features will probably not be as much fun as a small, well-implemented game. Of course, the most important part is to get the game working at all; you will be penalized harshly if the game is not playable, not matter how many other features you attempted to add.

Hints

- You may find it helpful to know what the best strategy for this game is, and you will probably it difficult to determine the best strategy for the game on your own. For example, knowing the best strategy might help you with deciding how many trees to include in a game, coding a computer opponent, or with determining an appropriate win condition for the pyromaniac (if you decide to include one). The CMUMC book linked in the footnote includes a solution. I'll give you a few hints below, but feel free to look up the strategy for the game online, or to discuss it with me. (Needless to say, don't look up anyone else's code for the game!)

- If both players follow best strategy, a game with more than 6 trees can take a long time to complete, and the required number of moves generally grows exponentially with the number of trees. You may want to keep this in mind when choosing the default number of trees in the game.

- As mentioned above, the pyromaniac's win condition should **not** be based on setting all the trees on fire. This results in an uninteresting game where both players play defensively, and nobody can win under optimal play (except in trivial cases).
- If the firefighter plays optimally, he will eventually win the game. If you decide to add a win condition for the pyromaniac, you can decide whether to preserve this property or not: the win condition can be strict enough that the pyromaniac will only win if the firefighter makes a mistake, lenient enough that the pyromaniac can always win as long as he doesn't make a mistake, or somewhere in between, where the outcome under best play depends on the starting position.