AP Computer Science Final Project - README Template

Instructions:
The first step in creating an excellent APCS final project is to write up a README. At this stage, this README file acts as your **project proposal**. Once you've filled in all components, Shelby will read through it and suggest edits. Ultimately, you need a document that adequately describes your project idea and **we must agree on this plan**.

Have one member of your group **make a copy of this Google Doc**. Then, they should share it with all other members **and with Mr. Shelby** so that every group member has edit permissions, and Shelby can add comments on your ideas.

There's a lot of parts of this document that you might not have full answers for yet. Because you haven't written the program yet, it's difficult to think about the **instructions** or **which group members will do which parts**. Even though this is hard to think about, you must have something in these sections that acts as your current plan. However, during the course of the project, you'll **continuously update this document**. This means that you will not be *held* to exactly what you put here - components of this document can change (and it's pretty common!).

There is one exception: the **Features List** section. Once Shelby OKs your README, the Features List section **cannot be modified**. For this reason, it is most important that you get a solid idea of what you want to make and the primary features it will have *now*.

Talk with your group. Consider drawing some pictures of what you think your project might look like. Be precise. When you're ready, fill this out together. Each component in brackets below ([these things]) should be replaced with your ideas. Note that there are several sample READMEs posted on this assignment for you to use as guidance.

----------When README is finalized, remove everything above this line----------

Polygon Protection

Authors: Antonio Cuan, Justin Yen
Revision: 5/13/2022

Introduction:
[In a few paragraphs totaling about ½ page, introduce the high-level concept of your program. What this looks like depends a lot on what type of thing you are making. An introduction for an *application* will look different than one for a *game*. In general, your introduction should address questions like these:
What does your program do?

What problem does it solve? Why did you write it?
What is the story?
What are the rules? What is the goal?
Who would want to use your program?
What are the primary features of your program?]

Enemies are attacking your area! Defend yourself by building automatic turrets that shoot incoming enemies. (If we cannot implement this feature, delete this:)[However, don't let your turrets do all the work for you. Go out to battle with your weapon to defeat the enemy.] The enemies come in waves that increase in difficulty. You gain money as the game progresses. Use the money to build more turrets and upgrade them. Can you survive long enough to complete the game?

Instructions:
[Explain how to use the program. This needs to be **specific**:
Which keyboard keys will do what?
Where will you need to click?
Will you have menus that need to be navigated? What will they look like?
Do actions need to be taken in a certain order?]

The player will have an indefinite amount of time to place turrets before a wave of enemies comes. The player starts the game with a set amount of gold. They can use gold to buy turrets in the side panel. The player can select a turret from the side panel by clicking on it. Then, the player can move their mouse to the open space and click where they want to place the tower. Placing a tower will also purchase it, meaning that the player will spend gold. The player can click a button to start the enemy wave when they are ready. The player repeats this process for multiple waves of increasing difficulty.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER):
Must-have Features:
[These are features that we agree you will *definitely* have by the project due date. A good final project would have all of these completed. At least 5 are required. Each feature should be fully described (at least a few full sentences for each)]

- The player can place turrets on the area that automatically fire at enemies. The turrets track the enemies as the enemies move (auto-aim).
- There are enemies that try to cross to the other side of the area (screen) and take damage from turrets.
- The enemies have pathfinding and go around turrets.
- The game prevents the player from placing turrets in a way that completely blocks enemies from going to the other side of the area (screen).
- The player gains money as the game progresses to buy turrets.
- When a certain number of enemies reach the other side of the area (screen), the game ends (the player loses), and the player can restart the game.

Want-to-have Features:
[These are features that you would like to have by the project due date, but you're unsure whether you'll hit all of them. A good final project would have perhaps half of these completed. At least 5 are required. Again, fully describe each.]

- A menu with a title start button. (This is a want to have because a menu is not strictly necessary for a working game.)
- The turrets have limited ranges.
- The enemies come in waves that increase in difficulty.
- The enemies have visible health bars.
- The player can control a character that can move around and attack enemies.
- The player can upgrade their turrets.
- The player has multiple different turrets to choose from that do different things.
- There are turrets that fire moving projectiles.
- There are multiple enemies types that act differently and have different properties.
- The player can choose different weapons for their controllable character that do different things.
- The player can upgrade their controllable character's weapon.

Stretch Features:
[These are features that we agree a *fully complete version of this program would have, but that you probably will not have time to implement*. A good final project does not necessarily need to have any of these completed at all. At least 3 are required. Again, fully describe each.]

- There are sound effects that play during the game, including sound effects for the turrets shooting, the enemies taking damage, and the player attacking enemies.
- There is original music that plays during the game.
- All game elements are original sprites that are pixel art or are hand-drawn
- There is a leaderboard so that you can compare your performance to other players'.

<u>Class List</u>:
Main
DrawingSurface
Screen
GameScreen
EndScreen
ScreenSwitcher

ScreenElement
Grid
Store

GameElement
Tower

Enemy
Projectile
PlayerCharacter
Weapon

Credits:
[Gives credit for project components. This includes both internal credit (your group members) and external credit (other people, websites, libraries). To do this:

- List the group members and describe how each member contributed to the completion of the final program. This could be classes written, art assets created, leadership/organizational skills exercises, or other tasks. Initially, this is *how you plan on splitting the work*.
- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]

Antonio:

- Drafted README
- Helped make UML diagram
- Set up project
- Coded initial code for Main and DrawingSurface classes
- Coded underlying structure for using the grid
- Coded breadth first search pathfinding algorithm for enemies
- Coded grid highlighting when play hovers mouse over the grid
- Coded selecting and placing towers in the grid
- Coded the start next wave button

Jusin:

- Helped draft README
- Helped make UML diagram
- Coded initial project classes and structure
- Coded basic graphics
- Coded underlying structure for enemy movement
- Coded attacking for towers (and projectiles)
- Coded gold and base health tracking

Outside resources:

- Processing library