**What the problem is:**

In my project I plan to create a platformer game that involves a lot of problem solving and logical reasoning, similar to some well-known games including: ‘Trap Adventure 2’, the ‘Super Mario’ series, and ‘Dead Cells’. All three of these games are kinds of platformer games. In ‘Trap Adventure 2’ and games from the ‘Super Mario’ series, the gameplay is mainly focused on parkour and evasion of obstacles to win. Both of these involve a life system where if the player is defeated or fails too many times, their progress is reset. In ‘Super Mario’ games and ‘Dead Cells’, completing certain objectives such as beating levels, or defeating enemies rewards you with score points or some form of reward or currency that can be used later on to benefit the player. In these games, completing a level brings you to the next, more difficult level, once all levels are completed the game ends and the player may replay if they wish to.

To achieve my task, my game will need to have the following: a character that can be controlled by the player; multiple levels (each increasing in difficulty); different types of platforms, such as disappearing and moving ones; obstacles in the levels, such as walls or blocks, and different types of these, like static obstacles that are inanimate and ones that are fake (e.g. pretend to be static, but may chase the player once character gets too close); lives and score system; a timer that is indicated not by a value, but in the background design of the game, the sun which is in the background to indicate between beginning and finish (day for beginning, night for end); collectibles and obtainable items; enemies and a final boss; settings/menu system; tutorial; and a menu where reward/currency and obtained items can be used.

**Stakeholders:**

The game’s graphic design will be very similar to the three games mentioned earlier; arcade and retro style. This art style is simplistic and useful for my game because it ensures that players are more focused on the core gameplay rather than how the game looks. This is important since the point of the game is to improve problem solving skills, if the player is too distracted by a beautiful graphic design, they won’t be fully concentrated on solving the problems that are presented in the game. By having the player’s focus set on the core gameplay itself, the game will be able to more accurately assess a player’s current set of logical reasoning skills, and improve them more efficiently. The game is also intended for computer and laptops, this is because PC keyboards allow for flexible controls. Furthermore, a keyboard allows for a better approach to improving problem solving because if I were to implement mobile controls for example, I would have to use icons or decals to indicate what each control does, and this doesn’t really help the player in developing their mind because they will easily be able to tell what each control does, whereas forcing the player onto a keyboard where they don’t know what each key does, allows them to think about how they can use the keys to beat the game.

Considering these factors, the game will be most suitable for audiences in the age group of 7 years to 18 years. This is because since the game is going to be available for PC only, and children of these ages are in school where computers are used more often, they will be able to play. The majority of children in this age group also likely already know the alphabet, so the keys on a PC keyboard will be understandable making them easier to understand which key does what (which will be explained in the tutorial of the game). Also, since some text is present in the game, the audience should be capable of basic reading which this age group should be capable of.

**How and why the problem is solvable via computational methods:**

The problem is solvable using computational methods because of the fact it is a game. Since the game is a platformer type of game, it will require inputs (to recognise what the player is doing so it can be drawn onto the screen or responses by the game can be made, especially movement and actions). When a player plays the game, the game also has to be interactive with the player since otherwise, the player is just making inputs and getting nothing out of it, which wouldn’t really mean they are playing a game, this means the game also has to output things to the player and runs calculations to tell what the player is doing. These require algorithms to do and therefore allows the problem to be solved via computational methods. The fact that the game has a form of repetition to it (due to levels, timers, scoreboards etc) also means that algorithms need to be used repeatedly to solve this problem.

**Thinking Concurrently**

By thinking concurrently, I can identify issues with the current solutions and fix them. This is important since I know I will have a scoreboard and timer (a simple scoreboard and the sun that rises and falls to indicate timer), therefore I already know that I need these two features to be updated at the same time since as the player is obtaining score, the timer is also moving.

**Thinking Ahead**

This way of thinking allows me to decide and plan ahead what I will do. This is important since if I do everything concurrently, there wouldn’t have been much time to properly think out what the best ways of accomplishing my goals are. It provides proper organisation so that when I do get to designing, and creating code, everything will be well organised so that I know exactly what to do, when, and how. This simply makes things easier for me since if I have to both think and do at the same time, I am more likely to make errors. Some things I plan on doing or using are:

* Using python; pygame to write my game as it is good for producing these type of retro games, especially since mine is pretty much all 2 dimensional and pygame is good for drawing 2 dimensional things.
* I know what base controls (standard controls) will be used such as the keys ‘w’, ‘a’, ‘s’, and ‘d’, which are very commonly used as movement controls in modern video games. This can be changed via settings that will be made later. This is also more comfortable for players that have already played platformer games before.
* Some outputs by the game are like sound effects to indicate player’s actions that caused something, such as buttons that can be interacted with,

**Thinking Abstractly**

**Thinking Logically**

**Thinking Procedurally**