Game Description:

<u>Title: Math Slayer</u>

Features of the Game:

1. Level-based Gameplay: Math Slayer features an unlimited number of levels, each increasing in difficulty. Players progress through the levels by defeating monsters and earning money.

2. Calculation-Based Combat: Players engage in battles with monsters by solving mathematical calculations within a time limit. Correct answers deal damage to the monster, while incorrect or slow responses result in damage to the player.

3. In-Game Store: Every 5 levels, players can access a store where they can recover their health and purchase items using the money earned from defeating monsters.

4. Increasing Difficulty: As players advance through the levels, the mathematical calculations become progressively more challenging, ranging from simple addition and subtraction to more complex two to three-digit calculations.

Levels	Operators (+/-/*)	digits
1-5	+	1
6-10	+	2
11-15	-	1
15-20	-	2
21-25	*	1
26-30	*	2
31-35	All	1
36-40	All	2
41-45	All	1
46-49	All	2
>50	All	2-3

5. Here are the details about the game:

Player initial hp	100
Monster initial hp	110
Player initial attack	10
Monster initial attack	12
Time limits	10 seconds
Coins earned every 5 levels	100

How to Play the Game:

1 Player

1. Start by creating a character with initial health points(hp) and attack.

2. Enter the game world and navigate through levels, encountering monsters along the way.

3. Engage in combat with monsters by solving mathematical calculations presented by the game within a specific time limit. Input the answer and hit enter before the timer runs out.

4. Correct answers cause damage to the monster, while incorrect answers or running out of time result in the player taking damage.

5. Defeat monsters to earn money and progress to the next level.

6. After every 5 levels, visit the in-game store to recover HP and purchase helpful items to aid in future battles.

7. Continue exploring and advancing through increasingly difficult levels, using strategy and quick thinking to defeat monsters and become a math champion.

2 Player

1. Player 1 will take a movement first.
2. Players only can attack by solving mathematical questions. Answering correctly within the time limits will counter an attack on the opponent. On the other hand, answering wrongly or running out of the time limit will cause the player to be attacked by the opponent.
3. Answering correctly will have a chance to drop items that will help players to win the game.
4. The probability of items dropping is as below

hp+10	5%
attack+10	16%
Revive stone	40%
Nothing drop	39%

Object-Oriented Concepts*:*

Object-oriented concepts can be utilized in the development of Math Slayer to organize and structure the game's code. For example:

1. Player Class: This class can encapsulate attributes like HP, armor, and the countdown timer. It can also include methods for dealing with player actions such as attacking, taking damage, and purchasing items.

2. Monster Class: This class can store monster attributes such as health points and damage values. It can also have methods for monster actions like attacking the player and calculating damage.

3. Store Class: This class can manage the in-game store, including functionalities such as recovering HP and purchasing items. The items class is composited by several classes so that the items can be accessed when the game is running.

4. Question Class: This class is used to generate the questions. Different types of questions can be generated by the class for the use of other classes by passing the function using composition.

Role of Linked Lists/Stacks/Queues:

Linked lists, stacks, and queues can be employed to manage various aspects of the game:

1. Linked Lists: A linked list can be used to maintain the sequence of levels in the game. Each level can be represented as a node in the linked list, with a reference to the next level. This allows for easy traversal and progression through the levels as the player completes each one. Linked lists are particularly useful for dynamic data structures where levels can be added or removed during gameplay.

2. Stacks: Stacks can be used to implement the combat system in the game. When a battle begins, the game can use a stack to store the mathematical calculations that need to be solved. Each calculation can be pushed onto the stack, and the player needs to solve them in a last-in-first-out (LIFO) manner. This ensures that the player solves the most recent calculation first, simulating real-time combat.

3. Queues: Queues can be used to manage the in-game store functionality. When the player reaches a store every 5 levels, the store can have a queue that stores items and health recovery options. The player can then interact with the store by dequeuing items from the front of the queue, simulating a first-in-first-out (FIFO) mechanism. This ensures fairness in accessing the items and maintains the order in which they were added to the store.

By utilizing these data structures, the game can effectively manage the game progression, combat system, and in-game store mechanics, providing an engaging and challenging experience for the players.