**Final Project Proposal**

**Group Name:** Albert Wen and Co.

**Groups Members:** Kami Beckford, Storm Lin, Maya Mansingh, Albert Wen

**Game Description:**

Name: Mapquest

When the game is first opened, the logo and instructions appear on the screen. Once the player clicks the mouse, a map appears with the character at the center. The player uses the arrow keys to move the character around the map. When the character hits certain locations, instructions on how to open a minigame appear at the top of the screen. Once the instructions are followed (once the player presses z on the keyboard), the game's title and instructions appear, along with more instructions on how to continue to the game. The game opens when those instructions are followed. After the player either wins or loses the game, they are returned to the main screen, and there is a little bit of interactive dialogue that responds to the player at the top of the screen.

The object of the game is to collect a number of keys that will unlock a temple. The keys are obtained by winning the minigames. The game is nonlinear, so the character can visit the locations in a random order. Once the player has won a game, they cannot return to it.

One game is a card matching game, where the player must turn over cards to find a match. Once a pair is found, it is eliminated from the options (removed from the array). One the array is empty (meaning all the cards have been matched), the game is over.

Another game is a dodging game, where the character is at the bottom of the screen. The mouse controls the character's movement. The objective is to not get hit by the objects falling from the top of the screen. If the character gets hit, the player loses one point. The player only receives an item if his or her final score is above negative five, meaning the character can only get hit by less than five objects in thirty seconds order to receive a key.

The third game is a mouse clicking game. The player has ten seconds to click the mouse as many times as possible (using only the right click). The goal is to click at least fifty times.

The fourth game is a riddle game. The player is given a multiple choice math problem. If the player answers all three questions correctly, the character receives a key.

The final game is a balancing game, where the player must use the left and right arrow keys to keep a beam at an angle less than ninety degrees in the twenty seconds that they are allotted in an effort to try and prevent a weight from falling off.

**What makes this game an appropriate challenge for your group?**

- It is nonlinear

- Different mini games require different code, along with the challenge of integrating them all into the main code
- Inventory must be tracked
- Good knowledge of arrays needed

**Who is the target audience?**
Due to the simple nature of the game and the fact that this game can help improve memory, coordination, reaction time, and mathematical skills, this game's target audience is children in elementary school.

**Lead Programmer:** Storm
**Project Manager:** Maya
**UI and Graphics:** Kami
**QA Tester:** Albert

**Additional Information:**

The group was inspired by the raindrop game that was done during class. Our math game was inspired by riddles. We originally were going to have riddles and multiple choice answers to them, but decided against it because displaying the options to the answers could make the game too easy.

When we first started this project, our code was fairly simple. If we needed congruent rectangles at different locations, we made them individually. As we worked on the code, we made a class and array list for the rectangles.

Also, when we started, each minigame was under its own void setup and void draw. When we had to integrate each game into the main code, we couldn't have several voids setup and voids draw, so we created functions to run the void setup and void draw of each game. The bull game specifically not only has a class, but also a bull system that could be run by just calling upon bullsystem.run (bs.run). The rest of the games are called upon more efficiently with just the use of the gameScreen method. We also renamed the voids setup and draw of each game as void PuzzleDraw, for example.

Creating a variable for gameScreen also allowed the an easy switch between games, the main screen, the welcome screen, and the game over screen. Each game or screen is assigned a numerical value , and when that numerical value is reached/called upon, the designated screen appears.