**MIPS Dots and Boxes Project Report**

**Your Name Here**

**Program Description**

The Dots and Boxes program is relatively simple. The data is made up of three arrays, two to store the horizontal and vertical edges and one to store the already created boxes. It takes in three pieces on input from the player per turn, the x coordinate, the y coordinate, and the cardinal direction in which the new edge is to be placed. If north or south are chosen, a vertical line will be chosen and added to the vertical edge array and if east or west are chosen the same will happen for the horizontal edge array. Then the box counter will go through the two arrays and check if any boxes have been made through the creation of this new edge, and update the scores accordingly based on whose turn it is. Then once the player has done this the computer will make its turn and randomly add a line somewhere on the board. This cycle repeats until there are no more available moves and the game ends.

**Challenges**

One of the larger challenges the team faced was figuring out how to store the connections. Lots of complicated ideas were thrown around using a vertex-based system but when it came down to it an edge-based system became a lot simpler as it required less data than storing four separate connections to one vertex, instead only storing the connections themselves. It also made figuring out if a box existed much easier, requiring only 4 array calls to see if the four lines required existed.

**What I Learned**

Write this yourself.

**Algorithms and Techniques**

The algorithms and techniques used in this program largely have to do with array searching. Since the values we are dealing with are so small and unordered, complex searching algorithms were unnecessary and not really cost efficient to what we would need to do so incremental searching was used. For adding the lines, the algorithm was to check if the input data was valid, find the location of the line through some arithmetic, check if the line already exists, and add it to the appropriate array if it does not. For counting the boxes, two increments were used in a nested loop to check through the x and y coordinates of the horizontal and vertical arrays and to add to the box array. In this nested loop, if four specified array coordinates were all marked to exist and a box didn’t exist already in this specific portion of the array, a box would be added. While it is not the most efficient search algorithm available, it is the simplest to implement and is already extremely fast due to the small amount of data being processed.

**Contributions**

Write this yourself.