**Project Report**

**a).**

The program simulates the THREE IN A LINE game. The player who connect three pieces first will be the winner where three pieces are considered connected if they lie in the same line either vertically, horizontally or diagonally without having the other’s piece in between.

**b).**

I had a hard time to implement the function that checks the connection diagonally because the function is non-leaf and requires the manipulation of $sp. I overcome the challenge by debugging where I set different positions in the program to print the actual value of variable to be compared with the desired value to see where my program goes wrong. I repeated the process until I found that some arguments as well as return address in the function has been modified unintentionally. Then I realize it is a problem of $sp, where saving and loading procedure should be included, which I forgot initially.

**c).**

By doing the project, I got more practiced in using assembly language and more decent understanding of how things work in assembly language. Also, I became more familiar with functional programming in MIPS and various instructions that manipulate all kinds of different data types.

And I learned that if I were to redo the project, I will first give an overview of the program and determine the API that I need to structure the program instead of writing functions one at a time when I need it. I could have made the structure of the program way clearer by analyzing of how functions support each other to remove redundancy.

**d).**

I used three one dimensional character arrays to represent the three vertical columns in which each of them containing six characters.

***The workflow***

Start a game -> Choose who goes first -> dropping pieces (updating the grid) back and force between the user and computer until there is a winner, or a draw game is reached -> restart or not

***The winning criteria is implemented as follow***

Assume a piece from user is placed at (i, j), where 0=< i <3 and 0 <= j < 6.

**1). Row checking**

Check whether pieces at positions (0, j), (1, j), (2, j) are all belongs to the user.

**2). Column checking**

When j > 2, Check whether pieces at positions (i, j), (i, j - 1), (i, j - 2) are all belongs to the user.

**3). Top-left to Bottom-right diagonal checking**

Check whether there are three of the positions (i + 2, j - 2), (i + 1, j - 1), (i , j), (i - 1, j + 1), (i - 2, j + 2), are valid and pieces at these valid positions are all belongs to the user.

**4). Bottom-left to Top-right diagonal checking**

Check whether there are three of the positions (i - 2, j - 2), (i - 1, j - 1), (i , j), (i + 1, j + 1), (i + 2, j + 2), are valid and pieces at these valid positions are all belongs to the user.

5). Any of the checking that is true will end the game by announcing the corresponding winner.

The same procedure applies to the move by computer.

**e).**

All work done by me.

**f).**

A graphic based display will make the program much appealing.