Midterm Project-Connect 4

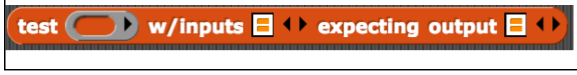
1. List of features:
   a. Drawing dots in "Connect Four style" – moving left, right and space keys to manipulate the dots. Implementing if-else statements so that a dot can be drawn under the following parameters [1) if tile is empty and 2) tile is located at the bottom of row or tile below already as a filled in dot]
   b. Declaring the winner - Implementing an algorithm that continuously attempts to detect a winning connect-four pattern every turn and informs the players who won the game.

2. The Game code is complex. First, we had to find the distance between each tile horizontally and vertically. So when we press the right, left and space key the chips will be dropped to the exact position of an empty tile. Then we had to stamp the already moved chip and change the costume it so it would look like if it's the other players turn. When we changed the costume, the chip had to return to the original position (x: 4.75; y: 145.25) which was just above the first tile of the 4th column (which is the center column of the board). After that, we implemented distinct algorithms for distinct cases in which the chip had to be dropped to an empty tile. These algorithms would vary depending on if there were already tiles on the board and if the last tile of the column you would place the chip in is empty. Finally, we had to implement 2 blocks that would check if player 1 won the game or if player 2 won the game. Both were similar, with the difference that for player one, we compared the conditions with the number 1 and with player 2 we compare the conditions with the number 2. Inside both blocks we implemented algorithms that checked the board for vertical patterns, horizontal patterns, and diagonal patterns.

3. We created a board that is a global variable made of a list of lists. The columns and the rows are also lists which are global variables that change each time we press space. The color is a global variable and we set it to a value of 1 at the beginning of the game and when our sprite changes costume the color changes to a value of 2. Then each time we change the costume the color is updated to a 1 or a 2 depending on what it was before pressing space. When we fill the board with chips, we are, instead, filling it with 1's or 2's. 1= red; 2= black. We use the script variables "column number" and "row number" in the block "move chip down". We set column number to a block called "image to column "d.position" that gives us the column number in the board where we are about to place our chip. This is useful because we update our column to find the column of the column number filled up with 0, 1 or 2's. With the script variable "row number" it give us the row number and later on the whole row filled up with 0, 1 or 2's after our chip moves down. Another important script variable is "counter" which can be find on "player 1 wins" or "player 2" wins blocks. This variable helps us find if a player won with a vertical or horizontal pattern by adding the counter by 1 if the next item of the column/row is a 1 or 2 (depending on which chip you are). Finally if the counter is equal to 4 it means that there are 4 chips of the same color in a row.

Eduardo Huerta-Mercado
MJ Harris

Block/function name	**Domain/Inputs**	**Range/Outputs**	**Behavior**
Start game `start game`	N/A	N/	Places a chip on initial position
Move chip down `move chip down`	N/A	N/A	Moves chips down each time I press space
Decide let positions of chip `decide left position of chip`	N/A	N/A	Move initial chip to left each time I use left arrow
Decide right position of chip `decide right position of chip`	N/A	N/A	Move initial chip to right each time I use right arrow
Player 1 wins `player 1 wins`	Lists	Booleans	Checks if player 1 haves 4 chips together (vertically, horizontally or diagonally)
Player 2 wins `player 2 wins`	Lists	Booleans	Checks if player 2 haves 4 chips together (vertically, horizontally or diagonally)
Item row, col of `item , of` board	Numbers, lists	Numbers	Gets item C, R of my board
Get column C of `get column of` board	Numbers, lists	List	Gives me the column C of the board as a list
Get row R of `get row of` board	Numbers, lists	List	Gives me the row R of the board as a list
Image to `image to column` column	Numbers	Numbers	Gives me the number of column where my chip is going too be placed
Image to row `image to row`	Numbers	Numbers	Gives me the number of row where my chip has been placed
Update column of board `update c of`	Numbers, lists	Lists	Update Column C of board filled up with 0, 1 or 2's
Update row of board `update r of`	Numbers, lists	Lists	Update row R of board filled up with 0, 1 or 2's
board `board`	N/A	Lists	Creates a 7x6 board made of ceros (list of lists)
Copy of board `copy of`	Lists	Lists	Makes a copy of the board
Set item row, column of board to value `set item , of to`	Numbers, Lists	Number	Changes the item C, R of my board to X

Test 	Lists	Booleans	Test if my blocks work.