Project Report

* 1. The program is a simplistic representation of the game Reversi, which is played between the user and a computer opponent. The display uses a Bitmap and input is received through Mars’s console. A menu prompts the user to play the game, then the game is run, when the game is over the result (won/lose) is displayed and the program exits.
  2. One of the main challenges that I faced when creating the validation section of the code, was in determining how to check all the possible directions to see if the move was valid and somehow store the valid endpoints so that if the move was valid the correct pieces could be flipped later. I solved this by creating an array and storing those endpoints if it was valid and a -1 if it was not. Also, when flipping the pieces, keeping track of the score was a slight problem as multiple correct directions caused the common pivot space to be counted multiple times, which I solved by checking if the pivot spot was already flipped, and if so skipping that flip. The main problem that was faced when creating the bitmap board, was coming up with the numbers to stop the loops for each piece for the given dimensions.
  3. I learned a lot about subroutines, loops, and if statements (branching), especially on base cases and how to end/skip statements. I also got a good understanding of how the Mars bitmap works, both with pixels coloring and picture representation.
  4. For validation, the starting X Y pieces were passed into a function along with the directional values for each of the 8 directions, separately, i.e. up is 0 for x and 1 for y. The spots between the edge of the board and the given X Y are checked until either: a same color piece (as player) is found and that piece’s X Y value are returned, a different color pieces is found and the loop is run again, an empty space is found or the edge is reached and a -1 is returned to show direction does not have a valid move. Also if a valid spot is returned it is checked to make sure that it is not next to the player chosen piece. If a valid move was chosen, then another function is called to flip the pieces, which simply starts at the chosen X Y and runs in each direction to the spot stored in the array of end pieces while keeping score of the number of pieces flipped. Displaying the board and pieces simply uses a loop that colors in each pixel, initially in a checkerboard pattern at the start and then with the pieces each time a player moves.