Project Report

1. Description of the program:

A card matching game, where the cards are pairs of a number and a multiplicative expression of that number.

1. The challenges that you and your partner had and how did the team overcome them:

Some challenges we faced included communication, division of labor, and certain technical challenges with the randomization program. We overcame communication by setting clear meeting times where we would meet in person, and by commenting our code well. We overcame division of labor by dividing our assignments into modules, where we would then come together if we were having problems or when we would connect their implementations with one another. We solved our randomization program by implementing a MIPS version of a well-known shuffling program with randomized indices, the reason this was a problem was that my partner had tried many other solutions before this.

1. What you have learned by doing the project:
   * 1. With focus and attention, one can solve problems in interesting and unique ways.
     2. Comments are an absolute necessity when working in a team, to improve readability (obviously) and productivity. Productivity is improved in that for a team member to edit or add functionality to some module comments are helpful in helping a coder to understand a program in a shorter amount of time so that they can add their own code earlier/faster.
2. A discussion of algorithms and techniques used in the program, e.g. how to display the board, how to check and remember a move from the user? How does the program work?
3. The board is displayed through a system of .asciiz arrays that are iterated through loops each row printing the board and printing the cards that are flipped, keeping them flipped if the match is correct.
   * These ascii arrays of rows aren’t edited/overwritten with the card values, instead another ascii array is kept separate that holds the other sides of the cards/their values.
   * For the cards that are to be permanently flipped we implemented a flag array called flippedCards which holds 1 in the positions that will always print the cards as flipped.
     + The flag array is given 1 values under MatchPrint which is called when the main checks that the values are equal, each value is returned via locationCheck.
4. We used a temporary print to print the user’s decision if it was wrong.
5. The program works on 4 .asm files, main.asm, locationCheck.asm, board.asm, and DataRand.asm.
   * main.asm prompts the user for card inputs, makes the card checks with subroutine calls to other files, and exits the program when the game is completed, by keeping a count from 8-0.
   * locationCheck.asm checks the location of a card and returns the integer value of the card from the input row and column, adjusting for randomized positions into the default value-position layout.
   * board.asm prints the board temporarily for incorrect decisions, changes the board permanently for correct decisions, and prints the current board while waiting for the next decision. And prints the flipped cards via an internal subroutine that takes from an array of the ascii faces of the card’s numbers and math, after adjusting for the randomized positions back to the default index positions of the array mentioned above.
   * DataRand.asm randomizes a .word array with a set of indices 0 - 15 to return to main and offset the cards to randomize the board. The array that is returned to main is saved into a save register ($s3) that is accessed by the appropriate subroutines/functions. For the shuffling of the randomized indices, we used the Fisher-Yates Shuffle algorithm.
6. Contributions of your partner (peer evaluation):
7. Carlos did most of the work for the randomization file
8. Helped implement the randomized index array into locationCheck.asm
9. Any suggestions you may have (optional):

Maybe for the extra credit functionality, i.e. the GUI and sound, suggest some documentation to become informed on how those that would want to do that could proceed initially?