1 Instructions

In this assignment, you will be required to write JavaScript functions that simplify playing of the variation of UNO

1.1 Data File Specification

An example of properly formatted file is shown in Figure 1. The first file encodes a deck, the second file encodes the discard pile, and the third file encodes the hands.

test02.deck.uno

```
r,6;r,r;g,5;y,4;b,0;g,6;g,s;g,4;y,1;g,d;g,0;r,r;b,2;r,0;y,9;g,2;g,3;
   b,3;y,5;b,2;r,8;b,d;g,s;g,d;g,9;w,d;y,5;y,2;r,4;y,8;b,8;r,3;r,s;r
   ,5;y,3;b,8;g,1;g,r;r,8;g,9;y,4;y,1;b,1;b,d;r,4;g,7;b,s;r,s;r,7;y,
   r;y,6;b,9;y,s;r,1;b,1;w,d;w,-;y,6;b,7;y,7;g,6;y,d;y,9;w,-;y,d;b,r
   ;w,d;b,4;r,6;w,-;g,7
```


test02.discard.uno

```
g,3;g,1;g,r;g,5;b,5;b,3;b,7;b,4;b,6;b,5
```


test02.hands.uno

```
r,9;r,d;y,7;y,s;y,3;w,-
r,7;r,d;g,8;g,2;y,0
g,8;b,6;r,9;y,2;g,4;b,s
w,d;r,1;y,r;r,2;r,5
r,3;y,8;r,2;b,r;b,9
```


2 One Player, One Move

The first part (`onePlayerOneMove` in the file `csce322h0mework02part01.js`) will take in two (2) arguments (a deck and a discard pile) and return a function that takes in one (1) argument (a

hand), and returns the deck, discard, and hand that is the result of the player in possession of the hand playing a card. The precedence for playing a card is as follows:

1. Extend a Wild Draw 4 if that is the most recently played (left-most) card (behind a `r,-`, `g,-`, `b,-`, or `y,-`)
2. Extend a Draw 2 if that is the most recently played card (not behind a `r,-`, `g,-`, `b,-`, or `y,-`)
3. Play the left-most card that matches the color of the most recently played card in the discard pile
4. Play the left-most Wild Draw 4
5. Play the left-most card that matches the symbol of the most recently played card in the discard pile
6. Play the left-most Wild
7. Draw (add to the back of the hand) the left-most card in the deck

If playing a Wild (or Wild Draw 4), the next player needs to know which color to play next. If the hand still contains non-Wild cards, play the card `c,-` (where `c` is the color of the left-most non-Wild card remaining in the hand). If the hand only contains Wild cards, play `r,-`. If the hand has been emptied (and, therefore, the game won), play `-,-`.

In the event that you need to extend a Wild Draw 4 or Draw 2 and cannot, you must draw all of the cards (possibly) built up from other players extending Wild Draw 4s or Draw 2s. If a Draw 4 was played, it would be directly behind a `r,-`, `g,-`, `b,-`, or `y,-`. Once you draw the number of cards required (or the size of the deck, whichever is smaller), place a copy of the a `r,-`, `g,-`, `b,-`, or `y,-` from the front of the discard pile on the front of the discard pile. This will let future players know that they must continue with a card of that color.

3 One Player, Many Moves

The second part (onePlayerManyMoves in the file csce322h0mework02part02.js) will take in two

(2) arguments (a deck and a discard pile) and return a function that takes in one (1) argument (a hand), and returns the deck, discard, and hand that is the result of the player in possession of the hand playing as many cards in a row as they can before emptying their hand or being unable to continue playing cards. The same rules for precedence of moves as onePlayerOneMove applies.

4 Many Players, One Move

The third part (manyPlayersOneMove in the file csce322h0mework02part03.js) will take in two (2) arguments (a deck and a discard pile) and return a function that takes in one (1) argument (a vector of hands), and returns the game that is the result of n turns being taken for a game with n players. The same rules for precedence apply, but `skip` and `reverse` cards will have these effects: if Player p plays a `reverse` on turn t, Player $p-1$ will take turn $t+1$ (or Player n will take turn $t+1$ if Player 1 played the reverse) assuming the turns are proceeding in ascending order. If turns are proceeding in descending order Player $p+1$ will take turn $t+1$ (or Player 1 will take turn $t+1$ if Player n played the reverse). If Player p plays a `skip` on turn t, Player $p+2$ will take turn $t+1$ (or Player 1 will take turn $t+1$ if Player $n-1$ played the skip, or Player 2 will take turn $t+1$ if Player n played the skip) if turns are proceeding in ascending order. If turns are proceeding in descending order, Player $p-2$ will take turn $t+1$ (or Player $n-1$ will take turn $t+1$ if Player 1 played the skip, or Player n will take turn $t+1$ if Player 2 played the skip).

5 Many Players, Many Moves

The fourth part (manyPlayersManyMoves in the file csce322h0mework02part04.js) will take in two (2) arguments (a deck and a discard pile) and return a function that takes in one (1) argument (a vector of hands), and returns the game that is the result of a game being played to its conclusion. Instead of n players combining to take n turns, turns will be taken following the rules of `manyPlayersOneMove` until a player empties their hand or the player whose turn it is cannot continue the game (either by playing a card or drawing a card from the deck).

6.1 helpers.js

A file named `helpers.js` has been provided with the functionality to read the `.uno` files into matrices. If a modified `helpers.js` file is not included with your submission, the default will be used in its place.