Joshua Adams
Exam 1
CS331

Part A

1. With the use of the car and cdr functions, I can virtually split the list in two pieces, the first element (2-List) and the rest of the list. This will allow me to look only at the first list and do some sort of operation required on it, and combine it later with the rest of the list.

2. If I do an operation, and then call the function recursively on the rest of the list, it will in turn look at each element (2-List) of the list. This will allow for me to do the operation necessary on each item (2-List) within the list.

3. You can combine it with the list that you are currently looking at, thus making it one big final list with the correct values that you want

4. The recursion will bottom out when the list that you are looking at is empty. This means that you are at the end of your list and there are no more elements to look at.

5. With a simple cons call, with recursion, we can combine the list that we are looking at, with the function called on the rest of the list, to give us the final, completed, reversed 2-Lists.

Part B

```
var flipflop = function(list){
	if(isNull(list)){
		return []; //reached end of list, return empty list
	} else {
		return cons(cons(car(cdr(car(list))),cons(car(car(list)),[])), flipflop(cdr(list)));
		//first arg of the cons is flipping the two elements of the 2list
		//second arg of the cons is recursively calling flipflop on the rest of the full list
	}
}
```

Part C

```
var flippyfloppy = function(list){
	return underscore.map(list, function(x) { return cons(car(cdr(x)),cons(car(x),[])); } );
	//the second arg of the map function returns what the first arg of the original recursive
	//function would give
}
```