CSC 425/525 Programming Assignment #1
Due: Monday, February 6

Remember: undergraduates may work in groups of up to 3 students

The jewels problem is stated as follows. Given a 3x3 grid of jewels, select specific jewels with your magic wand to change that jewel and its adjacent jewels (above, below, left, right) from the current jewel to the next jewel in the sequence. Jewels transition from diamond to ruby, ruby to emerald, and emerald to diamond. Your goal is to translate a given 3x3 grid (the start state) to a specified goal state by determining which jewels to tap with your wand and in which sequence. For instance, given the grid on the left, we might want to reach the grid on the right.

```
        D D R        E E E
        D D D ➔      E E E
        D D D        E E E
```

By touching the middle jewel, you transition from the state on the upper left to the following state.

```
        D R R
        R R R
        D R D
```

By next touching the upper left jewel, you transition from the above state to the following state.

```
        R E R
        E R R
        D R D
```

Number the grid entries as 0-8 (upper left hand corner is 0, 0-1-2 are the top row, etc). One solution from the above start state to goal state is the sequence of 0 – 3 – 7 – 8 – 2.

In this assignment, you will solve the jewel problem first by depth-first search and then by best-first (heuristic) search, using a heuristic function that you develop. As it is possible to revisit states in your search, you will want to maintain a closed list that stores each state already visited to indicate a dead-end. Rather than searching your closed list to see if the current state has already been visited you might instead use an array of Booleans where you convert the state into an array index. Since each array location can store one of three values and there are 9 grid locations, the array will need 3^9 (19683) total storage locations.

Once your search has found a solution, output it as the sequence of moves to make in order from start state to goal state (this may require reversing the order as indicated on the run-time stack). For instance, one solution to the above problem is 0 – 3 – 7 – 8 – 2. This should be output in that order. Also output the total number of states visiting during the search. There may be several solutions from any given start state to a goal state, only output one of them. Your solution may take more steps than 5 depending upon how you implement your search.

After you have successfully implemented the depth-first search version of the program, implement a best-first search including whatever heuristic function you came up with. Your heuristic does not necessarily have to be a *good* heuristic. Output the first solution found along with the number of total states visited. If this number is greater than that of your depth-first search, try to improve your heuristic (this is not necessary, but something you might want to try).

Use the earlier start and goal states to test your program.

Next, run it on the following three pairs of start and goal states:

D D D	E E R
D D D	E R E
D D D	R E R

D D D	R R E
D D D	R E R
D D D	E R R

D D D	R D R
D D D	D R D
D D D	R D R

Hand in:

1. Your source code for both implementations (depth-first, best-first)
2. Your output when running on the 3 sets of initial/goal states above (the output should be the sequence of grid locations you touch with your wand and the number of states visited during the search)
3. A short report that describes who in your group implemented which portions (if you worked in a group), what you learned from the assignment, how easy or challenging the assignment was
4. An explanation of your heuristic and whether you felt it improved the search and whether a better heuristic might help