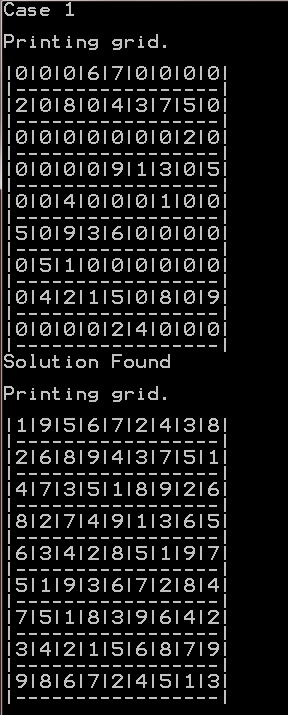
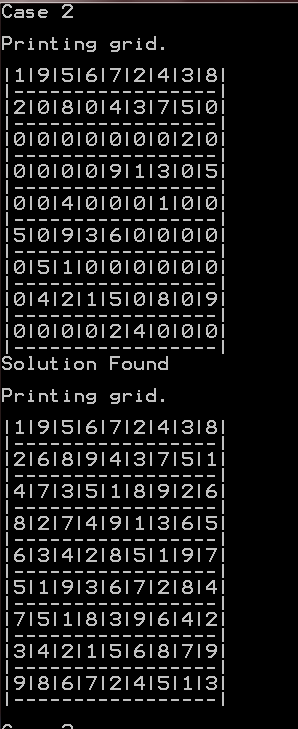
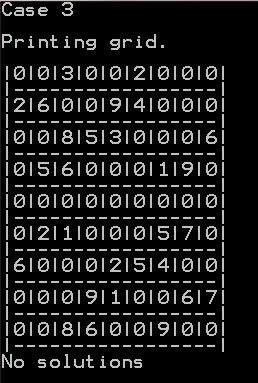
Sudoku Project 4 Report

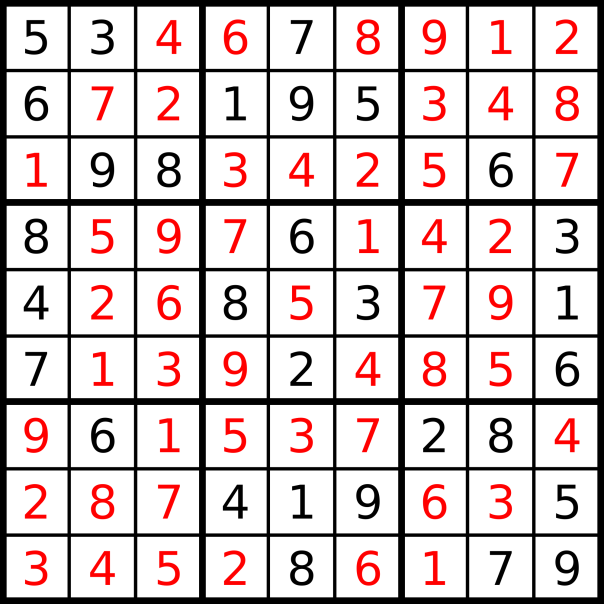
Test Cases:

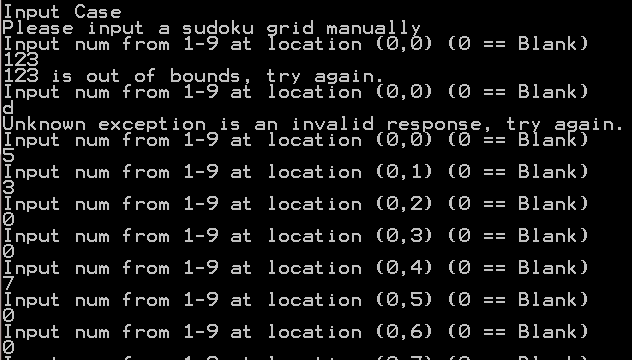






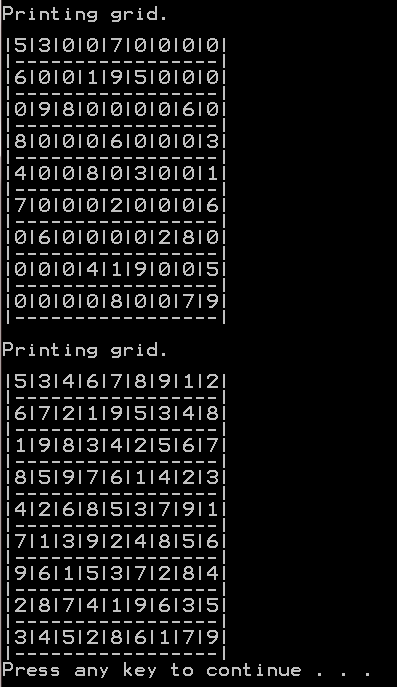
Input case:





…

Note: I added 1 to coordinate later such that the coordinates go form (1,1) to (9,9)



Discussion:

o Discussion of the used data structures.

The data structure used is a 9 by 9 array. This is not the most optimal array. One could argue that a one dimensional linked array such as the one used in the matrix example would have been better for memory space. However the 9 by 9 was easier to deal with in this program. Also since Sudoku has a preset dimension, the two dimensional array is sufficient because resetting the value from 0 to the number is less complexity than the insertion method of a one dimensional linked list.

o Discussion of the used algorithm.

This algorithm is a brute force trial and error method that uses backtracking when the guessing number is equal to a number in the row, column or box.

o Testing results with screenshots.

See above

o Screenshots of the results for the three problems set shown on page 2.

See above

o Discussion of the performance of the algorithm based on the results.

This algorithm has many worst case scenarios where the “rabbit-hole” recursion comes back with bad results. The backtracking method would require many trials.

o Discussion of weaknesses of the used algorithm and any possible enhancements.

Brute force is not a good method. In order to improve this method we need to think how humans would solve this problem. People will not typically go one at a time and plug in numbers from one to nine. We can lower the complexity by initially ordering which number occurs most often can trying to pin point where that number can fit in. Then when this method fails, brute force can supplement. I believe this method would be better because this is how Sudoku is solved by normal people.