Homework 3 Experiments:
We ran our CSP solver on various test files:
australia_coloring.txt represented the Australia map coloring problem. It had the following constraints:
SA ne WA
SA ne NT
SA ne Q
SA ne NSW
SA ne V
WA ne NT
NT ne Q
Q ne NSW
NSW ne V
T lt 3
SA lt 3
WA lt 3
NT lt 3
Q lt 3
NSW lt 3
V lt 3

This problem is solved in trivial time with or without forward checking.
SOLUTION:
NSW: 2
NT: 2
Q: 1
SA: 0
T: 0
V: 1
WA: 1

Forward checking actually does nothing in this case, because of the nature of the problem and AC3:
trying SA = 0
domains BEFORE AC_3: {'WA': [0, 1, 2], 'Q': [0, 1, 2], 'T': [0, 1, 2], 'V': [0, 1, 2], 'SA': [0], 'NT': [0, 1, 2], 'NSW': [0, 1, 2]}
domains AFTER AC_3: {'WA': [0, 1, 2], 'Q': [0, 1, 2], 'T': [0, 1, 2], 'V': [0, 1, 2], 'SA': [0], 'NT': [0, 1, 2], 'NSW': [0, 1, 2]}
trying Q = 1
domains BEFORE AC_3: {'WA': [0, 1, 2], 'Q': [1], 'T': [0, 1, 2], 'V': [0, 1, 2], 'SA': [0], 'NT': [0, 1, 2], 'NSW': [0, 1, 2]}
domains AFTER AC_3: {'WA': [0, 1, 2], 'Q': [1], 'T': [0, 1, 2], 'V': [0, 1, 2], 'SA': [0], 'NT': [0, 1, 2], 'NSW': [0, 1, 2]}

In another test, called nine.txt, we had 9 variables (A1-A9) that had to be assigned to numbers 1-9 distinctly (Like one of sudoku). Here are the constraints:

A3 eq 1
A1 eq 5
A1 gt 0
A1 lt 10
A2 gt 0
A2 lt 10
A3 gt 0
A3 lt 10
A4 gt 0
A4 lt 10
A5 gt 0
A5 lt 10
A6 gt 0
A6 lt 10
A7 gt 0
A7 lt 10
A8 gt 0
A8 lt 10
A9 gt 0
A9 lt 10
A1 ne A2
A1 ne A3
A1 ne A4
A1 ne A5
A1 ne A6
A1 ne A7
A1 ne A8
A1 ne A9
A2 ne A1
A2 ne A3
A2 ne A4
A2 ne A5
A2 ne A6
A2 ne A7
A2 ne A8
A2 ne A9
A3 ne A1
A3 ne A2
A3 ne A4
A3 ne A5
A3 ne A6
A3 ne A7
A3 ne A8
A3 ne A9
A4 ne A1
A4 ne A2
A4 ne A3
A4 ne A5
A4 ne A6
A4 ne A7
A4 ne A8
A4 ne A9
A5 ne A1
A5 ne A2
A5 ne A3
A5 ne A4
A5 ne A6
A5 ne A7
A5 ne A8
A5 ne A9
A6 ne A1
A6 ne A2
A6 ne A3
A6 ne A4
A6 ne A5
A6 ne A7
A6 ne A8
A6 ne A9
A7 ne A1
A7 ne A2
A7 ne A3
A7 ne A4
A7 ne A5
A7 ne A6
A7 ne A8
A7 ne A9
A8 ne A1
A8 ne A2
A8 ne A3
A8 ne A4
A8 ne A5
A8 ne A6
A8 ne A7
A8 ne A9
A9 ne A1
A9 ne A2
A9 ne A3
A9 ne A4
A9 ne A5
A9 ne A6
A9 ne A7
A9 ne A8

With and without forward checking, we got a solution:
A1: 5
A2: 2
A3: 1
A4: 4
A5: 3
A6: 7
A7: 6
A8: 9
A9: 8

However, forward checking is faster. Here are how the domains get reduced as it makes assignments:

domains BEFORE AC_3 {'A1': [5], 'A3': [1], 'A2': [2], 'A5': [3], 'A4': [4], 'A7': [4, 6, 7, 8, 9], 'A6': [4, 6, 7, 8, 9], 'A9': [4, 6, 7, 8, 9], 'A8': [4, 6, 7, 8, 9]}
domains AFTER AC_3 {'A1': [5], 'A3': [1], 'A2': [2], 'A5': [3], 'A4': [4], 'A7': [6, 7, 8, 9], 'A6': [6, 7, 8, 9], 'A9': [6, 7, 8, 9], 'A8': [6, 7, 8, 9]}
trying A7 = 6
domains BEFORE AC_3 {'A1': [5], 'A3': [1], 'A2': [2], 'A5': [3], 'A4': [4], 'A7': [6], 'A6': [6, 7, 8, 9], 'A9': [6, 7, 8, 9], 'A8': [6, 7, 8, 9]}
domains AFTER AC_3 {'A1': [5], 'A3': [1], 'A2': [2], 'A5': [3], 'A4': [4], 'A7': [6], 'A6': [7, 8, 9], 'A9': [7, 8, 9], 'A8': [7, 8, 9]}
trying A6 = 7

For full sudoku, not using forward checking will run for a VERY long time. With forward checking, a solution is found in about 10 seconds.

Our tests our included in this zip folder.