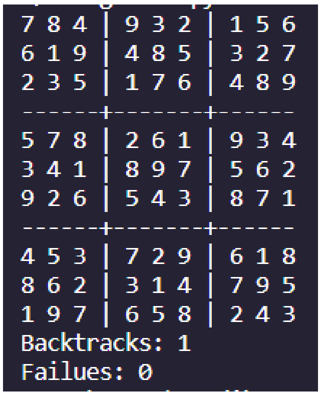
Ferdinand Schumacher Ytteborg & Fredrik Hægermark

TDT4136 – Exercise 4

a) and b)

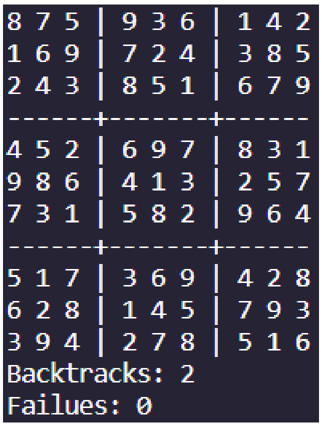
Solutions to the four sudoku boards that were provided are shown below. Starting with board (a), (b), (c) and finally (d). The number of backtracks and backtrack failures are also shown.

Task a)



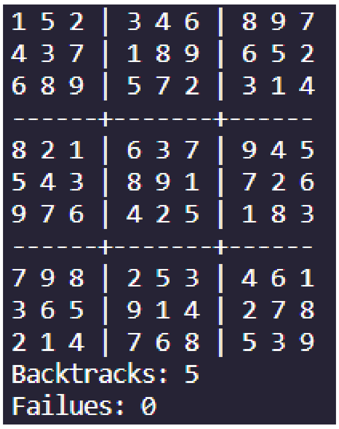
```
7 8 4 | 9 3 2 | 1 5 6
6 1 9 | 4 8 5 | 3 2 7
2 3 5 | 1 7 6 | 4 8 9
------+-------+------
5 7 8 | 2 6 1 | 9 3 4
3 4 1 | 8 9 7 | 5 6 2
9 2 6 | 5 4 3 | 8 7 1
------+-------+------
4 5 3 | 7 2 9 | 6 1 8
8 6 2 | 3 1 4 | 7 9 5
1 9 7 | 6 5 8 | 2 4 3
Backtracks: 1
Failues: 0
```

Task b)



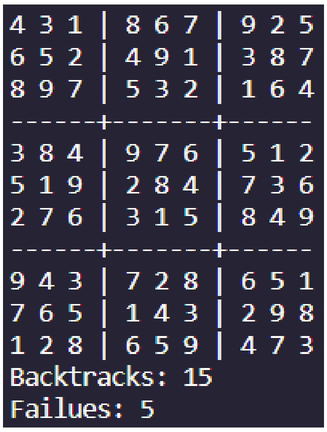
```
8 7 5 | 9 3 6 | 1 4 2
1 6 9 | 7 2 4 | 3 8 5
2 4 3 | 8 5 1 | 6 7 9
------+-------+------
4 5 2 | 6 9 7 | 8 3 1
9 8 6 | 4 1 3 | 2 5 7
7 3 1 | 5 8 2 | 9 6 4
------+-------+------
5 1 7 | 3 6 9 | 4 2 8
6 2 8 | 1 4 5 | 7 9 3
3 9 4 | 2 7 8 | 5 1 6
Backtracks: 2
Failues: 0
```

Task c)



```
1 5 2 | 3 4 6 | 8 9 7
4 3 7 | 1 8 9 | 6 5 2
6 8 9 | 5 7 2 | 3 1 4
------+-------+------
8 2 1 | 6 3 7 | 9 4 5
5 4 3 | 8 9 1 | 7 2 6
9 7 6 | 4 2 5 | 1 8 3
------+-------+------
7 9 8 | 2 5 3 | 4 6 1
3 6 5 | 9 1 4 | 2 7 8
2 1 4 | 7 6 8 | 5 3 9
Backtracks: 5
Failues: 0
```

Task d)



```
4 3 1 | 8 6 7 | 9 2 5
6 5 2 | 4 9 1 | 3 8 7
8 9 7 | 5 3 2 | 1 6 4
------+-------+------
3 8 4 | 9 7 6 | 5 1 2
5 1 9 | 2 8 4 | 7 3 6
2 7 6 | 3 1 5 | 8 4 9
------+-------+------
9 4 3 | 7 2 8 | 6 5 1
7 6 5 | 1 4 3 | 2 9 8
1 2 8 | 6 5 9 | 4 7 3
Backtracks: 15
Failues: 5
```

c)

By starting to compare the number of backtracks and backtrack failures for the different boards, we can see that the number increases with the difficulty level, as one might expect intuitively. Backtracking algorithm uses depth-first search and choses one variable at the time until there are no more options. Then it backtracks. Algorithm is called recursively; the performance is heavily dependent on implementing good heuristics for the next unassigned variable and ordering of domain values.

In our algorithm no heuristics are used. In our implementation the algorithm takes the next available unassigned variable without sorting. Performance might have been improved by sorting the number of legal values ascendingly, such as a MRV-heuristics. Could for instance been done by implementing a least-constraining-value heuristic. Would have been interesting to see how this might have improved the performance of the algorithm by looking at backtracks and backtrack failures.

It might be worth mentioning that for the first board the AC3 algorithm finds the solution on the first tree. This is opposite of the hardest one, where the algorithm has to “test out” different values for unassigned variables and repeat the inference multiple times. In other words some dead ends and failed backtracks.