CSCE 312 Computer Organization **10/11/2015**
Gustavo Estrela de Matos 925003559

Problem 1

1 -

2 -

```
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 747 nanoseconds
 The measured code took 0 seconds and  4294966847 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 653 nanoseconds
 The measured code took 0 seconds and  4294966954 nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 775 nanoseconds
 The measured code took 0 seconds and  4294966785 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 699 nanoseconds
 The measured code took 0 seconds and  4294966942 nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 624 nanoseconds
 The measured code took 0 seconds and  4294966953 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 716 nanoseconds
 The measured code took 0 seconds and  4294966881 nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 728 nanoseconds
 The measured code took 0 seconds and  4294966837 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 592 nanoseconds
 The measured code took 0 seconds and  4294967047 nano seconds to run
```

3-

```
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 742 nanoseconds
 The measured code took 0 seconds and  4294966816 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 548 nanoseconds
 The measured code took 0 seconds and  4294967107 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 638 nanoseconds
 The measured code took 0 seconds and  4294966959 nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
./sol3 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 708 nanoseconds
 The measured code took 0 seconds and  4294966868 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 581 nanoseconds
 The measured code took 0 seconds and  4294967029 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 717 nanoseconds
 The measured code took 0 seconds and  4294966888 nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
./sol3 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 1061 nanoseconds
 The measured code took 0 seconds and  4294966790 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 1538 nanoseconds
 The measured code took 0 seconds and  4294966213 nano seconds to run
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 751 nanoseconds
 The measured code took 0 seconds and  429496[5874] nano seconds to run
gustavo-xps:ex1\$ make run
./sol1 < input.txt > output.txt
./sol2 < input.txt >> output.txt
./sol3 < input.txt >> output.txt
cat output.txt
2
Timer Resolution = 1 nanoseconds
 Calibrartion time = 0 seconds and 676 nanoseconds
 The measured code took 0 seconds and  4294966896 nano seconds to run
2
Timer Resolution = 1 nanoseconds
```

The performance of the algorithms were similar, but it was possible to see that solution 1 was the fastest and the solution 2 was the slowest.

4 – Solution 1 was the fastest because it has the fewest number of conditions and applications of masks. The solution from lab1 has a similar number of conditions, but applies masks more than the solution 1, which probably was the reason this solution was slower than the fastest. Solution 2 was the slowest, probably because it has too many conditions, because in the worst case it can verify all the possible input values.

5 – The problem in my solution is that it applies masks too many times in the input variable. Possible solutions could be changing the way the conditions are verified, to something similar to solution 1, or using variables to keep the result of masks (it wouldn't be a good solution because it would cause waste of memory).

Problem 2

1 – SPST means Single Pole, Single Throw. NO – normally open.

Problem 3

1 – Assuming that actuators are enabled for one device at time we can use the negation of the lines to determine which device is being turned off.

Address bus lines				**I/O Enable**	**Actuator control lines**							**Comment**
A3	**A2**	**A1**	**A0**		**A/c**	**D0**	**D1**	**D2**	**D3**	**Engine starter**	**Wipers**	
X	X	X	X	0	0	0	0	0	0	0	0	Nothing happens
0	0	0	0	0	0	0	0	0	0	0	0	Nothing happens
0	0	0	0	1	1	0	0	0	0	0	0	A/c get turned on
0	0	0	1	1	0	1	0	0	0	0	0	Door 1 gets locked
0	0	1	0	1	0	0	1	0	0	0	0	Door 2 gets locked
0	0	1	1	1	0	0	0	1	0	0	0	Door 3 gets locked
0	1	0	0	1	0	0	0	0	1	0	0	Door 4 gets locked
0	1	0	1	1	0	0	0	0	0	1	0	Engine starts
0	1	1	0	1	0	0	0	0	0	0	1	Wipers turned on
0	1	1	1	1	0	0	0	0	0	0	0	Nothing happens
1	0	0	0	1	0	1	1	1	1	1	1	A/c get turned off
1	0	0	1	1	1	0	1	1	1	1	1	Door 1 unlocked
1	0	1	0	1	1	1	0	1	1	1	1	Door 2 unlocked
1	0	1	1	1	1	1	1	0	1	1	1	Door 3 unlocked
1	1	0	0	1	1	1	1	1	0	1	1	Door 4 unlocked
1	1	0	1	1	1	1	1	1	1	0	1	Engine stops
1	1	1	0	1	1	1	1	1	1	1	0	Wipers turned off
1	1	1	1	1	1	1	1	1	1	1	1	Nothing happens