Problem set 2, Lexical analysis and parsing – Magnus Halvorsen

Task 1

a)

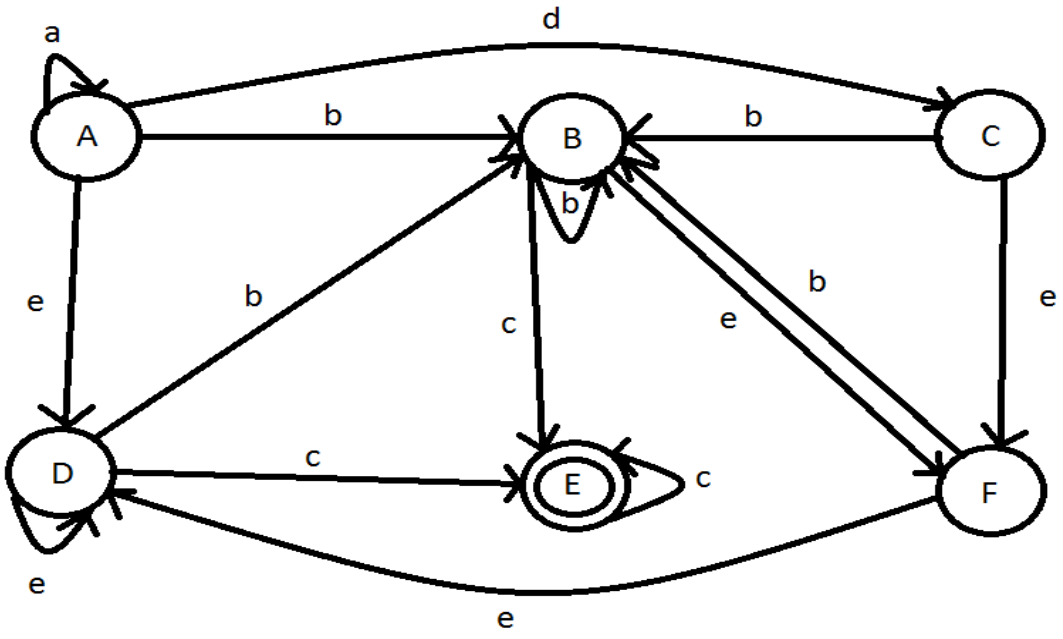
Original NFA:

	a	b	c	d	e	epsilon
1	{1,2}	-	-	-	-	{4}
2	-	{2, 3}				
3	-	-	{6}	-	-	-
4	-	-	-	{2}	-	{5}
5	-	-	-	-	{3}	{2}
6	-	-	-	-	-	-

Converted into DFA:

NFA states	DFA states	a	b	c	d	e
{1, 2, 4, 5}	A	{A}	{B}	-	{C}	{D}
{2, 3}	B	-	{B}	{E}	-	{F}
{2}	C	-	{B}	-	-	{F}
{2, 3, 5}	D	-	{B}	{E}	-	{D}
{3, 6}	E	-	-	{E}	-	-
{2, 5}	F	-	{B}	-	-	{D}

Illustration of the DFA:



b)

A language which wants an equal, but variable, amount of different symbols can not be described by a regular expression. E.g. you want a fixed number of a's, followed by an equal number of b's. Regular expressions are not defined to accept such a language, because the size of the DFA required to recognize such a language is unbounded. In other words, it becomes really impractical, unless we have an unlimited amount of memory.

Task 2

a) Ambiguous grammar is grammar that can be represented by more than one parse tree. I.e. it's a grammar that produces more than one leftmost or rightmost derivation of the same sentence. It's problematic because it causes programs to become undeterministic, i.e. your program may provide different output for the same input. This will result in buggy, unpredictable and unusable programs.

b)

I can not find more than one derivation for each example expression I've tried. In addtion the grammar is representing reverse-polish notation, which does not suffer from the ambiguity-problems of infi notation. Thus my conclusion is that the grammar is not ambiguous.

c) Left recursive grammar is grammar where the leftmost symbol is the same as the left side of the production. E.g. S -> Sa | epsilon. This is problematic for top-down parsers for the following reason: When the parser finds an something that matches the the example production, e.g. a, it will call the production S->Sa. But since the lookahead doesn't change until a terminal in the body is matched, the input remains the same. Thus the parser will call the production S->Sa again, and again – and we end up with a endless loop.

d)

Got stuck on debugging the programming part, so haven't finnished the theory. Will deliver it later, if this is not sufficent to get the assignment accepted. I was also not able to finnish debugging the code part, so could not get it running.