SOLUTION NOTES

Compiler Construction 2001 Paper 3 Question 3 (AM)

This is the file as submitted to the Examiners in January 2001.

(a)

```
S -> 0 T | 1 T
T -> 0 T | 1 T | . U | .
U -> 0 U | 1 U | e N | 0 | 1
E -> e N
E -> 0 N | 1 N | 0 | 1
```

(16 rules)

(b) (without using ϵ RHS's, but this would be OK too)

```
N -> 0 T | 1 T | 0 | 1
S -> N . N | N . | N . N e N | N . e N
```

(8 rules).

(c) (i) identical: firstly aaS or $aaaS$ means any number of a's greater than 1; so the first two rules in T code the first three rules in S. The last three rules of T code the last two rules in S.

(ii) both generate strings not generated by the other, e.g. only T generates adc and only S generates $aaaadcc$.

(d) For lexical analysis. Lexer would appear as the first logical stage in compilation (char stream to token stream). Its likely interface to the rest of the compiler is a routine `void lex()` which gets the next token from the input stream, setting a variable `token` to the token just read. Auxiliary variables (perhaps more formally fields of subclasses of Token) hold the attribute of complex tokens (like the characters of a 'string' token).