p491
AM

Solution Notes

[Syllabus: "lexical and syntax analysis"]

(a) A context-free grammar is a 4-tuple (N, T, S, R) where N and T are disjoint sets of respectively non-terminal and terminal symbols, $S \in N$ is the start symbol, and R is a set of *rules* of the form $\alpha \to u_1 \dots u_k$ where $\alpha \in N$ and $u_1, \dots, u_n \in (N \cup T)$.

(b) The grammar is as above, the language it generates that subset of T^* (strings of tokens from T) which can be generated from S using a finite number of rules.

(c) The grammar is ambiguous because the string 1+1+1 is may be generated in two separate ways:

$$E \to E+E \to 1+E \to \dots \to 1+1+1$$

and

$$E \to E+E \to E+1 \to \dots \to 1+1+1.$$

(d) (N, T, S, R) where $N = \{E\}, T = \{1, 2, X, +, *, -\}, S = E$ and

$$R = \{E \to 1, E \to 2, E \to X, E \to E+E, E \to E*E, E \to -E\}$$

(e) (i)

```
E ::= F | E+F | E*F
F ::= -P | P
P ::= 1 | 2 | X
```

(e) (ii)

```
E ::= F | F+E | F*E
F ::= -P | P
P ::= 1 | 2 | X
```

(e) (iii)

```
E ::= F | E+F
F ::= -G | G
G ::= P * G | P
P ::= 1 | 2 | X
```

(f) I

```
ParseTree rdE()
    {   ParseTree x = rdF();
        while (token == "+") { nexttoken(); x = mkplus(x, rdF(); }
        return x;
    }
```

```
ParseTree rdF()
    {   if (token == "-") { nexttoken(); return mkneg(x, rdG(); }
        else  return rdG();
    }
ParseTree rdG()
    {   ParseTree x = rdP();
        if (token == "*") { nexttoken(); x = mktimes(x, rdG(); }
        return x;
    }
ParseTree rdP()
    {   ParseTree x;
        switch (token)
        {   case '1':
            case '2':
            case 'X': x = mkprimitivenode(token);
                      nexttoken();
                      return x;
            default: error("expecting 1/2/X");
        }
    }
ParseTree ReadSentence()
    {   ParseTree x = rdE();
        if (token != EOF) error("junk after program");
    }
```