1. (a)  

i. Left-most derivation:

\[
T \Rightarrow F \Rightarrow P \Rightarrow (T) \Rightarrow (F) \Rightarrow (P) \Rightarrow (I) \Rightarrow (DI) \Rightarrow (7I) \Rightarrow (7DI)
\]

\[
\Rightarrow (78I) \Rightarrow (78D) \Rightarrow (789)
\]

Sometimes left-most derivation steps are explicitly indicated by an \( \text{lm} \) subscript as follows:

\[
T \Rightarrow F \Rightarrow P \Rightarrow (T) \Rightarrow \ldots \Rightarrow (789)
\]

Note that if the goal is just to derive a word, any derivation will do, not just a left-most one (unless you’ve been explicitly asked to provide a left-most derivation as here, of course). However, being systematic and sticking to e.g. left-most derivations where possible can help in avoiding mistakes. Anyway, to illustrate that there are other possibilities, here is a right-most derivation of the same word:

\[
T \Rightarrow F \Rightarrow P \Rightarrow (T) \Rightarrow (F) \Rightarrow (P) \Rightarrow (I) \Rightarrow (DI)
\]

\[
\Rightarrow (DDI) \Rightarrow (DDD) \Rightarrow (DD9) \Rightarrow (D89) \Rightarrow (789)
\]

ii. Left-most derivation:

\[
T \Rightarrow T + T \Rightarrow F + T \Rightarrow P + T \Rightarrow I + T
\]

\[
\Rightarrow D + T \Rightarrow 7 + T \Rightarrow 7 + F \Rightarrow 7 + F * F
\]

\[
\Rightarrow 7 + P * F \Rightarrow 7 + N(A) * F \Rightarrow 7 + g(A) * F \Rightarrow 7 + g(T) * F
\]

\[
\Rightarrow 7 + g(F) * F \Rightarrow 7 + g(F * F) * F \Rightarrow 7 + g(P * F) * F
\]

\[
\Rightarrow 7 + g(I * F) * F \Rightarrow 7 + g(D * F) * F \Rightarrow 7 + g(3 * F) * F
\]

\[
\Rightarrow 7 + g(3 * P) * F \Rightarrow 7 + g(3 * I) * F \Rightarrow 7 + g(3 * D) * F
\]

\[
\Rightarrow 7 + g(3 * 5) * F \Rightarrow 7 + g(3 * 5) * P
\]

\[
\Rightarrow 7 + g(3 * 5) * (T) \Rightarrow 7 + g(3 * 5) * (F) \Rightarrow 7 + g(3 * 5) * (P)
\]

\[
\Rightarrow 7 + g(3 * 5) * (N(A)) \Rightarrow 7 + g(3 * 5) * (f(A)) \Rightarrow 7 + g(3 * 5) * (f())
\]

iii. It is not possible to derive \( 1 + 2 * 3 \). There are only two productions that introduce parentheses, and they always introduce them as balanced pairs. Thus it is not possible to derive a string with a single unmatched parenthesis as in this case.

iv. It is not possible to derive \( 1 + 7(9) \) because a number cannot be followed directly by a left (opening) parenthesis. The substring \( 7(9) \) must have been derived from \( T \). There are only two productions that introduce parentheses. The production \( P \rightarrow N(A) \) cannot have been used to derive \( 7(9) \) because, although \( P \) can be derived from \( T \), \( 7 \) cannot be derived from \( N \). The other possibility is the production \( P \rightarrow (T) \). But, as numbers are only derivable via \( I \), this would only be possible if a word \( IP \) can be derived form \( T \). This in turn is not possible as the \( I \) ultimately has to be derived from a \( T \), and inspection then shows that the only terminals that can follow an \( I \) are \( +, *, \) and \( ) \).

[Marking: 5, 10, 5, and 10 for a total of 30 marks]
(b) Derivation tree:

[Diagram of a derivation tree with labels and operations indicated.]
(c) Another derivation tree:

The fact that there are two different derivation trees for one word implies that the grammar is ambiguous.
[Marking: 5 marks]

2. (a) The following CFG $G_E$ is an unambiguous grammar satisfying the requirements. $G_E = (N, T, P, S)$ where:

- $N = \{E_1, E_2, E_3, E_P, I, I_T, D, D_1\}$
- $T = \{\_, =, \oplus, \otimes, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $P$ is given by:

\[
\begin{align*}
E_1 & \rightarrow E_2 = E_2 | E_2 \\
E_2 & \rightarrow E_2 \oplus E_3 | E_3 \\
E_3 & \rightarrow E_P \otimes E_3 | E_P \\
E_P & \rightarrow I \mid (E_1) \\
I & \rightarrow 0 \mid D_1 I_T \mid -D_1 I_T \\
I_T & \rightarrow D I_T \mid \epsilon \\
D & \rightarrow 0 \mid D_1 \\
D_1 & \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{align*}
\]

- $S = E_1$
[Marking: 25 marks]

(b) Derivation tree for $42 = 0 \otimes -10 \otimes (1 \oplus 7)$:
[Marking: 10 marks]
3. (a) The word \(ab\) is accepted. ID sequence:

\[
\begin{align*}
(q_0, ab, \#) & \vdash (q_0, b, 0\#) \\
& \vdash (q_0, \epsilon, \#) \\
& \vdash (q_1, \epsilon, \#)
\end{align*}
\]

Accepting configuration since \(q_1\) is an accepting state and since all input has been read.
[Marking: 3 marks]

(b) The word \(aababc\) is accepted. ID sequence:

\[
\begin{align*}
(q_0, aabacc, \#) & \vdash (q_0, ababcc, 11\#) \\
& \vdash (q_0, babcc, 011\#) \\
& \vdash (q_0, abcc, 11\#) \\
& \vdash (q_0, bcc, 011\#) \\
& \vdash (q_0, cc, 11\#) \\
& \vdash (q_0, c, 1\#) \\
& \vdash (q_0, \epsilon, \#) \\
& \vdash (q_1, \epsilon, \#)
\end{align*}
\]

Accepting configuration since \(q_1\) is an accepting state and since all input has been read.
[Marking: 7 marks]

(c) The word \(ac\) is not accepted. There are three possible moves initially:

\[
\begin{align*}
(q_0, ac, \#) & \vdash (q_1, ac, \#) \\
or
(q_0, ac, \#) & \vdash (q_0, c, 0\#) \\
or
(q_0, ac, \#) & \vdash (q_0, c, 11\#)
\end{align*}
\]

However, no further moves are possible in either of the first two cases, and neither resulting configuration is accepting (input remains in both cases).
In the third case, a move is possible:

\[
(q_0, c, 11\#) \vdash (q_0, \epsilon, 1\#)
\]

But at this point, the PDA is stuck again. Once more, this is not an accepting configuration (the state is not final and there is remaining input anyway).
We have now covered all possibilities, and we have thus shown that there is no sequence of IDs leading to an accepting configuration, which means that the word \(ac\) is rejected.
[Marking: 10 marks]