1. Let $G$ be the grammar

\[
S \rightarrow A \ a \ A \ b \mid B \ b \ B \ a \\
A \rightarrow \epsilon \\
B \rightarrow \epsilon
\]

a) Is $G$ LL(1)? Explain.
b) Is $G$ SLR(1)? Explain.


a) Construct all the different parse trees for $a - a + a$.
b) Construct an equivalent unambiguous grammar that makes the operators $+$ and $-$ left associative and of the same precedence.
c) Draw the parse tree according to your grammar for the input string $a - a + a$.
d) Construct a syntax directed definition using your grammar that maps infix expressions into abstract syntax trees (ASTs).
e) Show the AST that gets generated for the input string $a - a + a$.

3. a) Construct three-address code for the following program fragment. State what assumptions you are making.

```
prod = 0.0;
for (i = 0; i < 10; i++)
    prod = prod + a[i] * b[i];
```

b) Construct a flow graph for your three-address code.

4. Draw a block diagram showing the six phases of a typical optimizing compiler. Describe the representation of the program before and after each phase. Which phase takes the most time when compiling a program?
5. Lambda calculus

   a) Reduce the following lambda calculus expression into normal form using both normal order evaluation and applicative order evaluation:

   \[ \lambda z. \left( \left( \lambda x. (+ x 3) \right) \left( \lambda y. (* y y) (a) \right) \right) \]

   b) Let \( F \) be the lambda expression \( \lambda t. (\lambda x. t(x)) (\lambda x. t(x)) \). Evaluate the expression \( FG \) where \( G \) is any lambda expression. What sort of behavior does \( F \) exhibit?