Consider the compilation of the following (miniJava) expression for a register-oriented processor architecture like MIPS (here variables x and y are int variables stored in memory)

\[(x + y) * x + (x + y)\]

(a) Show an AST for this expression and use Sethi-Ullman numbering on each node to determine the minimum number of registers needed to evaluate the expression. How many registers are needed?

The AST nodes are labeled with the minimum number of registers following the Sethi-Ullman algorithm. Thus 3 registers are needed to evaluate this expression.

(b) Show the tuple code generated by simple code generation using temporaries.

(c) Show how the tuple code can be simplified using common subexpression elimination.
(d) Determine the lifetimes of each temporary \( t_1 \) – \( t_5 \). What is the minimum number of registers \( k \) needed to evaluate the tuple code above?

\[
\begin{array}{cccccc}
\text{t1} & \text{t2} & \text{t3} & \text{t4} & \text{t5} \\
\hline
\text{t1} := x & & & & \\
\text{t2} := y & & & & \\
\text{t3} := t1 + t2 & & & & \\
\text{t4} := t3 * t1 & & & & \\
\text{t5} := t4 + t3 & & & & \\
\end{array}
\]

The horizontal line test shows a minimum of two registers will be needed to hold the 5 temporaries.

(e) Show an assignment of temporaries to \( k \) registers by constructing the interference graph and coloring it using \( k \) colors.

The interference graph can be colored using \( k = 2 \) colors, so the expression in (a) can be executed using just 2 registers following common subexpression elimination:

\[
\begin{align*}
\text{R1} & := x \\
\text{R2} & := y \\
\text{R2} & := \text{R1} + \text{R2} \\
\text{R1} & := \text{R2} \ast \text{R1} \\
\text{R1} & := \text{R1} + \text{R2}
\end{align*}
\]