1 – Parsing Techniques (10 points)
Write out the LR(0) automaton for the following grammar:

1. S → E ;
2. E → E + T
3. E → T
4. T → id
5. T → id [ E ]
6. T → ( E )
2. Parsing Types (10 points)
Consider the two equivalent ways of declaring variables with their types:

**C Style:**
- integer `x;`
- `boolean *y;`
- `double z[5];`

**Pascal Style:**
- `x: integer;`
- `y: pointer to boolean;`
- `z: array of 5 double;`

For each of these declaration styles, write a complete grammar that can express any variable declaration. Make sure to include integer, boolean, and double as base types, allow for pointers and arrays, as well as arbitrarily recursive pointers to pointers, arrays of arrays, etc. Clearly indicate the grammar class of each style.
3. Intermediate Representations. (10 points)
Write a function that simplifies expressions with a common multiplier according to the algebraic distribution of multiplication: $a*b + a*c = a*(b+c)$. Your function should accept a valid struct expr * (as defined in the class project) and return a valid struct expr * equivalent to the first, changing only the form of expressions as above. You may write helper functions to simplify your solution. For example, if the input expression is:

$$x = y*z + y*(2+w)$$

then your routine should return this AST:

$$x = y*(z+(2+w))$$

```c
struct expr * simplify_expr( struct expr *e ) {
```
4 – Register Allocation (10 points)

Explain how a highly optimized compiler would allocate registers for the following code fragment on a three-address RISC architecture. Write out how the code would be expressed in any intermediate forms, and sketch the most important data structures used. Indicate what register (r1, r2, r3, …) would hold what values. Clearly state the minimum number of registers needed. Assume each variable (a, b, c, d, x, y, p, q) is a parameter to the enclosing function.

\[
\begin{align*}
    x &= (a+b)*(c+d); \\
    d &= (a+b)*(x+y) + x*y; \\
    x &= (a+b)*(c+d); \\
    a &= d*(x+y); \\
    \text{if}(a>3) \{ \\
        \quad \text{int } p = a*x*d; \\
        \quad \text{print } p; \\
    \} \text{ else } \{ \\
        \quad \text{int } q = a*x; \\
        \quad \text{print } q; \\
    \}
\end{align*}
\]
5 – Garbage Collection. (10 points)
Explain how to build a conservative garbage collector for a C program.