Tagging procedure:

\[
\text{(define (tagged-list? x tag)}
\quad \text{(and (pair? x) (eq? (car x) tag))})
\]

Problems

1. Build a tagged abstraction for variables:

\[
\text{(define *variable-tag* 'variable)}
\]

(a) Write the constructor \text{make-variable}:

\[
\text{(define (make-variable vname)}
\]

(b) Write the type predicate \text{variable?}:

\[
\text{(define (variable? x)}
\]

(c) Write the selector \text{varname}:

\[
\text{(define (varname var)}
\]

(d) Write the equality predicate \text{variable=?}:

\[
\text{(define (variable=? v1 v2)}
\]
Tagged abstraction for constants:

(define *constant-tag* 'constant)

(define (make-constant c)
  (list *constant-tag* c))

(define (constant? x)
  (tagged-list? x *constant-tag*))

(define (constval c)
  (if (constant? x)
      (cadr x)
      (error "not a constant: " c)))

Tagged abstraction for polynomials:

(define *poly-tag* 'poly)

(define (make-polynomial var terms)
  (list *poly-tag* var terms))

(define (poly? x)
  (tagged-list? x *poly-tag*))

(define (poly-get-var poly)
  (if (poly? poly)
      (cadr poly)
      (error "not a polynomial:" poly)))

(define (poly-get-term i poly)
  (if (poly? poly)
      (list-ref (caddr poly) i)
      (error "not a polynomial:" poly)))

(define (poly-get-terms poly)
  (caddr poly))

2. Write constant-add:

   (define (constant-add c1 c2)
3. Write a basic `add`, which works only on constants and polynomials, assuming you have a procedure `poly-add` which adds two polynomials:

```
(define (add exp1 exp2)
```

4. Draw a box-and-pointer diagram of the representation of $5x^2 + 3x + 1$.

5. Write `poly-add`, which adds two polynomials

   (a) First write `add-terms`, which takes two lists of terms and returns a new list of sum terms:

   ```
   (define (add-terms t1 t2)
   ```

   (b) Then write `poly-add` using `add-terms`:

   ```
   (define (poly-add p1 p2)
   ```
6. Write \texttt{var->poly}, which \textit{promotes} a variable to a polynomial:

\begin{verbatim}
(define (var->poly var)
)
\end{verbatim}

7. Write \texttt{const->poly}, which \textit{promotes} a constant to a polynomial:

\begin{verbatim}
(define (const->poly var c)
)
\end{verbatim}

8. Write \texttt{->poly}, which converts it’s input to a polynomial:

\begin{verbatim}
(define (->poly var exp)
)
\end{verbatim}

9. Write a new version of \texttt{add} which uses promotion. Use the following procedure to guess what variable to use when promoting:

\begin{verbatim}
(define (find-var e1 e2)
  (cond ((poly? e1)
    (poly-get-var e1))
    ((poly? e2)
      (poly-get-var e2))
    ((variable? e1)
      e1)
    ((variable? e2)
      e2)
    (else
      (make-variable 'x))))

(define (add exp1 exp2)
)
\end{verbatim}