```
(defun assignment-statement-p (x)
   (and (equal (len x) 1)
         (and (equal (len (car x)) 2)
               (first (car x)) (symbolp (first (car x)))
               (expression-p (second (car x))))))
```

```
(defun expression-p (x)
   (and (consp x)
         (or (load-expression-p x)
              (add-expression-p x)
              (xor-expression-p x)...)))
```

```
(defun add-expression-p (x)
   (and (equal (len x) 3)
         (equal (first x) 'add)
         (variable-or-numberp (second x))
         (variable-or-numberp (third x))))
```