0 Zdefiniuj funkcję *atom?* przyjmującą jeden argument.
Jeśli argument jest atomem funkcja zwraca `#t`, w przeciwnym przypadku zwracana jest wartość `#f`.

---

1 Napisz wyrażenie odpowiadające poniższemu stosując jedynie *let*.

```
((lambda (a b c op1 op2)
   (op1 (op2 a b) c))
 12 5 7 + *)
```

---

2 Zapisz poniższe wyrażenie korzystając jedynie z *let*.

```
(let* ((val 5)
       (val2 (- val))
       (val3 (+ val val2))
       (val4 (- val2 val3)))
  (- val (+ val val2) (- val2 val3)))
```

---

3 Funkcja *member?* zwraca wartość boolowską `#t` jeśli *atom* znajduje się w liście *lista* lub `#f` w przeciwnym przypadku. Dopisz brakujący fragment funkcji.

```
(define (member? atom lista)
  (cond
    ((null? lista) #f)
    ((eq? atom (car lista)) #t)
    (else (                          ))))
```

---

4 Funkcja *append* zwraca listę złożoną z elementów list *lista1* oraz *lista2*. Poniższa definicja zawiera błąd, wytłumacz na czym on polega. Przykład działania dla poprawnej funkcji
`(append '(a b c) '(1 2 3))` ⇒ `(a b c 1 2 3)`

```
(define (append lista1 lista2)
  (cond
    ((null? lista1) lista2)
    (else (cons (car lista1) (append (cdr lista1) (cdr lista2))))))
```

---

5 Funkcje *reverse1* oraz *reverse2* operują na liście *lista* zmieniając kolejność jej elementów.
`(reverse1 '(a b c))` ⇒ `(c b a)`
`(reverse2 '(a b c))` ⇒ `(c b a)`
Jakie wyniki uzyskamy dla listy `(a (b c d) e)`? Czy będą one identyczne dla funkcji *reverse1* i *reverse2*? Odpowiedź uzasadnij.

```
(define (reverse1 lista)
  (if (null? lista) '()
      (append (reverse (cdr lista)) (list (car lista)))))

(define (reverse2 lista)
  (cond ((null? lista) '())
        ((pair? lista)
         (append (reverse2 (cdr lista))
                 (list (reverse2 (car lista)))))
        (else lista)))
```

---

6 Funkcja *untitled* operuje na liście *lista*, w rezultacie działania zwraca pojedyńczą wartość liczbową. Przykładowe wywołanie
`(untitled '(a ((b) c)))` ⇒ `3`
Co reprezentuje zwrócona przez funkcję wartość?

```
(define (untitled lista)
  (cond
   ((not (list? lista)) 0)
   ((null? lista) 0)
   (else (max (+ 1 (untitled (car lista))) (untitled (cdr lista))))))
```

---

7 Jaka lista zostanie zwrócona przez poniższe wyrażenie?

```
(let ((x '(a dull boy jack))
      (y '(play makes no work))
      (z '(and all)))
(cons (cadr z)
(cons (cadddr y)
(cons (car z)
(cons (caddr y)
(cons (car y)
(cons (cadr y)
(cons (cadddr x)
(cons (car x)
(cons (cadr x)
(cons (caddr x) '()))))))))))))
```

---

8 Funkcja *untitled* operuje na liście *lista*, w rezultacie działania zwraca pojedyńczą wartość liczbową. Przykładowe wywołanie
`(untitled '(a ((b) c)))` $\Rightarrow$ `3`
Co reprezentuje zwrócona przez funkcję wartość?

```
(define (untitled lista)
  (cond
   ((not (list? lista)) 0)
   ((null? lista) 0)
   (else (max (+ 1 (untitled (car lista))) (untitled (cdr lista))))))
```

---

9 Dana jest funkcja *countme* zliczająca kolejno swoje wywołania.
Jaka wartość zostanie zwrócona po 7. wywołaniu funkcji *countme*?
Jaką wartość będzie przechowywać zmienna *counter* po 7. wywołaniu funkcji *countme*?

```
(define countme
  (let ((counter 0))
    (lambda ()
      (let ((retval counter))
        (set! counter (+ counter 1))
        retval))))
```

---

10 Dana jest funkcja *nvalval* przyjmująca jeden argument liczbowy n
Czy poniższa funkcja jest poprawna? Jeśli nie, zaproponuj lepsze rozwiązanie.
Jaką wartość otrzymamy w wyniku wywołania funkcji *nvalval* dla liczby *5*?

```
(define nvalval
  (lambda (n)
    (letrec ((nval (lambda (n) (+ n val)))
             (val 10))
      (+ (nval (* n n)) val))))
```
