Lab 3 – Lists and Symbolic Data

The **quote mark** ' treats an expression as literal data:
`'(+ 2 3)` evaluates to the list `(+ 2 3)`
`'(pizza with anchovies)` evaluates to the list `(pizza with anchovies)`
`'pizza` evaluates to the symbol `pizza`

car returns the first element of a list:
`(car '(strawberry ice cream soda))` → `strawberry`

cdr returns everything but the first element of a list:
`(cdr '(strawberry ice cream soda))` → `(ice cream soda)`

The **list** function creates a list:
`(list 'ice 'cream 'soda)` → `(ice cream soda)`
`(list 'one 'fish 'two 'fish)` → `(one fish two fish)`

cons adds a new element to the front of a list:
`(cons 'banana '(cream pie))` → `(banana cream pie)`
`(cons 'weird '())` → `(weird)`

null? checks if a list contains no elements:
`(null? '(banana cream pie))` → `#f`
`(null? '(banana))` → `#f`
`(null? '())` → `#t`

1. Write the function `double-first`, which takes a list of symbols as input and returns the same list with an extra copy of the first symbol on the front. Your function should behave as shown in the examples below:

```
(double-first '(hip hooray)) → (hip hip hooray)
(double-first '(rain go away)) → (rain rain go away)
(double-first '(green eggs and ham)) → (green green eggs and ham)
```

2. Write the function `remove-first-two`, which takes a list of symbols as input and returns the list with its first two symbols removed. Assume that the input list will always contain at least two symbols.

```
(remove-first-two '(Sam loves green eggs and ham)) → (green eggs and ham)
(remove-first-two '(peanut butter pancakes)) → (pancakes)
(remove-first-two '(ice cream)) → ()
```

3. Write the function `remove-second`, which takes a list of symbols and removes the second symbol.

```
(remove-second '(chocolate cream pie yum yum)) → (chocolate pie yum yum)
(remove-second '(hot texas chili)) → (hot chili)
(remove-second '(ice cream)) → (ice)
```

4. Write the function `swap-first-two`, which takes a list of symbols and exchanges the positions of the first two symbols.

```
(swap-first-two '(Sam loves green eggs and ham)) → (loves Sam green eggs and ham)
(swap-first-two '(this is really fun)) → (is this really fun)
(swap-first-two '(banana split)) → (split banana)
```

5. Write the function `remove-third`, which takes a list of symbols and removes the third symbol.

```
(remove-third '(Sam loves green eggs and ham)) → (Sam loves eggs and ham)
(remove-third '(this is not easy)) → (this is easy)
(remove-third '(ice cream soda)) → (ice cream)
```

6. Write the function `insert-third`, which takes a symbol and a list as input, and inserts the symbol in the third position of the list.

```
(insert-third 'fried '(Sam loves green eggs and ham)) → (Sam loves fried green eggs and ham)
(insert-third 'pie '(coconut cream)) → (coconut cream pie)
(insert-third 'two '(one fish fish fish)) → (one fish two fish fish)
```

7. Write the function `replace-third`, which takes a symbol and list as input, and replaces the third symbol of the list with the specified symbol.

```
(replace-third 'bacon '(Sam loves green eggs and ham)) → (Sam loves bacon eggs and ham)
(replace-third 'tomato '(spaghetti with meat sauce)) → (spaghetti with tomato sauce)
(replace-third 'cake '(iced cheese balls)) → (iced cheese cake)
```

8. Write the function `three-copies`, which takes a single symbol as input, and returns a new list containing three copies of the symbol.

```
(three-copies 'ha) → (ha ha ha)
(three-copies 'fun) → (fun fun fun)
```

9. Write the function `combine`, which takes two lists as input, each of which should contain exactly two symbols, and creates a single new list containing all of the symbols in the same order. The resulting list should not contain any inner parentheses.

```
(combine '(the cat) '(big fish)) → (the cat big fish)
(combine '(one two) '(three four)) → (one two three four)
```

10. Write the function `interleave`, which takes two lists as input, each of which should contain exactly two symbols, and creates a single new list with the symbols interleaved as shown in the examples below.

```
(interleave '(the cat) '(big fish)) → (the big cat fish)
(interleave '(black truffle) '(olive oil)) → (black olive truffle oil)
```

11. Write the function `at-least-one?`, which checks if a list has at least one element. Hint: use `null?`

```
(at-least-one? '(apple banana cherry)) → #t
(at-least-one? '(apple banana)) → #t
(at-least-one? '(apple)) → #t
(at-least-one? '()) → #f
```

12. Write the function `exactly-one?`, which checks if a list has *exactly* one element, returning #t or #f accordingly. Hint: if the list itself is not empty, check if its cdr is empty. You are not allowed to use the `length` function for this problem.

```
(exactly-one? '(apple banana cherry)) → #f
(exactly-one? '(apple banana)) → #f
(exactly-one? '(apple)) → #t
(exactly-one? '()) → #f
```

13. Write the function `at-least-two?`, which checks if a list contains at least two elements. You are not allowed to use `length` for this problem.

```
(at-least-two? '(apple banana cherry)) → #t
(at-least-two? '(apple banana)) → #t
(at-least-two? '(apple)) → #f
(at-least-two? '()) → #f
```

14. Write the function `at-least-three?`, which checks if a list contains at least three elements. You are not allowed to use `length` for this problem.

```
(at-least-three? '(apple banana cherry)) → #t
(at-least-three? '(apple banana)) → #f
(at-least-three? '(apple)) → #f
(at-least-three? '()) → #f
```

15. In the U.S., hours are usually specified as a number in the range 1-12 followed by either am or pm. Midnight is written as 12 am, and noon is written as 12 pm. In Europe, hours are usually specified as a number in the range 0-23, with no am or pm designation. Hour 0 corresponds to midnight, hour 12 to noon, and hours 13 through 23 to 1 pm through 11 pm.

    Write a function called `to-european-time` that converts American hours to European hours. Your function should take an hour number in the range 1-12 and a symbol `am` or `pm` as input. It should return a single number in the range 0-23. If the hour is outside the range 1-12, or a symbol other than `am` or `pm` is given, the function should return the symbol `invalid`. Try to write your function definition in a succinct way, using just a few lines of code. Test your function on at least the following inputs:

```
(to-european-time 12 'am) → 0
(to-european-time  1 'am) → 1
(to-european-time  2 'am) → 2
(to-european-time 11 'am) → 11
(to-european-time 12 'pm) → 12
(to-european-time  1 'pm) → 13
(to-european-time  2 'pm) → 14
(to-european-time 11 'pm) → 23
(to-european-time  0 'am) → invalid
```

16. Write a function called `to-american-time` that converts European hours to American hours. Your function should take an hour number in the range 0-23 as input. It should return a *list* containing the hour number and the symbol `am` or `pm`. If the hour is outside the range 0-23, the function should return the symbol `invalid`. Try to write your function definition in a succinct way, using just a few lines of code. Test your function on at least the following inputs:

```
(to-american-time  0) → (12 am)
(to-american-time  1) → (1 am)
(to-american-time  2) → (2 am)
(to-american-time 11) → (11 am)
(to-american-time 12) → (12 pm)
(to-american-time 13) → (1 pm)
(to-american-time 14) → (2 pm)
(to-american-time 23) → (11 pm)
(to-american-time 24) → invalid
```