Name of the Student: _____

Q 1

Write a recursive function `P-SUM`, which takes a list of two element lists of integers, e.g. `((7 8) (1 13) (4 1))`, and returns their sum, 34 for this case.

Q 2

Write a recursive function `L-PROD`, which takes an integer and a list of integers, returns a list of integers where each element in the second list got multiplied by the first integer argument.

Q 3

Write a recursive function `PAIR-PROD`, which takes a list of two element lists of integers as above, but this time returns a list of products of these pairs. E.g. an input like `((7 8) (1 13) (4 1))` should yield `(56 13 4)`.

Q 4

Write a recursive function `BLANK-N` , which takes an integer `N` and a list, and replaces every `N`th element of the list with the symbol `X`.

```
* (blank-n 3 '(1 3 4 5 1 5 4))

(1 3 X 5 1 X 4)
```

Q 5

Re-write `MULTI-MEMBER` and `REM-LAST` from the previous assignment, recursively and without using any built-ins like `MEMBER`, `REVERSE` or `LAST`.[1]

Q 6

Write a recursive function `C-FIND`, which takes an object and a list and returns the position it first encounters the element, 0 otherwise.

Q 7

Modify your solution to Q 6, so that it returns the position it *last* encounters the element, 0 otherwise. Don't use `REVERSE`, `MEMBER` or `LAST`.

Q 8

Write a recursive function `NSUB-LIST`, which takes two lists and returns `T` if the elements in the first can be found in the second in the same order as they appear in the first list without necessarily being continuous:[2]

```
* (nsub-list '(a b) '(a b))

T
* (nsub-list '(b a) '(a b))

NIL
* (nsub-list '(a b) '(c a c b c))

T
```

Q 9

Write `SUB-LIST`, which behaves like `NSUB-LIST` but it gives `T` only if it finds the first list *continuously* in the second.

[1] Try these without checking the solutions of the previous assignment. If you check, try to write these without checking the solution the other day.

[2] You can use `LENGTH`.