1. Given a list as a parameter, write a function `positive-count` that returns the number of positive numbers in the list; return `nil` if the list is empty or has any non-numbers.

2. Given the wind speed of storms: `((name-1 speed-1) ... (name-n speed-n))` as a parameter, write LISP functions `storm-categories` to generate category names (39-73 is Tropical-Storm, 74-95 is Hurricane-Cat-1, 96-110 is Hurricane-Cat-2, 111-130 is Hurricane-Cat-3, 131-155 is Hurricane-Cat-4, and 156 or higher is Hurricane-Cat-5) and `storm-distribution` to calculate the number of storms in each category. You may assume the speed values in the argument list are integers with value $\geq 39$.

```lisp
> (defconstant *storms2004* '((bonnie 65) (charley 150) (frances 145) (ivan 165) (jeanne 120)))
*STORMS2004*
> (storm-categories *storms2004*)
((BONNIE TROPICAL-STORM) (CHARLEY HURRICANE-CAT-4) (FRANCES HURRICANE-CAT-4) (IVAN HURRICANE-CAT-5))
> (storm-distribution *storms2004*)
((TROPICAL-STORM 1) (HURRICANE-CAT-1 0) (HURRICANE-CAT-2 0) (HURRICANE-CAT-3 1) (HURRICANE-CAT-4 2) (HURRICANE-CAT-5 1))
```

3. The `member` function doesn’t check the existence of an element in a nested list. Write a recursive function `nested-member` that returns `t` if the first argument appears in the second argument, which can be a nested list. The function returns `nil` otherwise. For example,

```lisp
> (nested-member 'b '(a (b c)))
T
```

4. Describe (in the comments) how you would use a list to represent a simple (inverted) family tree (no siblings) with ancestors toward the bottom of the tree. For example:

```
John / 
 Mark Mary / 
 James Jane Peter Pat / 
 GeorgeH Barbara Bill Hillary GeorgeW Laura Barack Michelle
```

Use your representation to define constant `*family-tree*`. Write the the `parents` and `grandparents` functions; for example:

```lisp
> (defconstant *family-tree* ...) ... 
> (parents *family-tree* 'Mary)
(PETER PAT)
> (grandparents *family-tree* 'John)
(JAMES JANE PETER PAT)
> (parents *family-tree* 'GeorgeH)
NIL
```

5. The Euclidean distance between two points, $A$ and $B$, is defined as $\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$, where $a_i$ and $b_i$ are elements of $A$ and $B$ in $n$ dimensions. Consider each point is represented by a list in LISP. Without using iteration or recursion, write the `euclidean` function with two parameters. Assume the two parameters have lists of the same length and only numbers in the lists. For example:

```lisp
> (euclidean '(1 2 3) '(4 5 6))
5.196152 ; return value, # of decimal places not important
```

6. Describe (in the comments) how you would use a nested list to represent a (traditional) family tree with ancestors toward the top. For example, in the following tree:

```
a+b / | | 
c+u d+v e+w f / | 
m+x n+y o p q 
r
```
a is married to (+) b and they have children c, d, e, and f. For each married couple (+), the second person is not part of the original family. Use your representation to define constant `*family-tree2*`. Write the `spouse`, `siblings`, `children`, `grandchildren`, `parents2`, `grandparents2` functions; for example:

```lisp
> (defconstant *family-tree2* ...) ... 
> (spouse *family-tree2* 'v) 
D 
> (spouse *family-tree2* 'p) 
NIL 
> (siblings *family-tree2* 'n) 
(M O) 
> (siblings *family-tree2* 'y) 
NIL 
> (children *family-tree2* 'b) 
(C D E F) 
> (children *family-tree2* 'v) 
NIL 
> (grandchildren *family-tree2* 'a) 
(M N O P Q) 
> (parents2 *family-tree2* 'p) 
(E W) 
> (grandparents2 *family-tree2* 'p) 
(A B)
```