Purely Functional Lists

Functional Data Structures are immutable. An object is immutable if its state cannot be modified after it is created.
Our objective is to create a functional data structure - an Immutable List supporting basic operations mentioned below. Some languages do support immutable lists. For this exercise, you are supposed to implement an Immutable List from scratch.

head - Returns the first element of a list.
tail - Returns a new list with all elements of the original list except the first.
cons - Takes an argument and prepends it to the list.
drop - Takes an integer ‘n’ as argument and returns a new list after removing first n elements from the list.
reverse - Returns the reverse of a list.

Each one of the above operations must return a new list with the input list left intact. The operations **head**, **tail** and **cons** may be used to implement the rest of the operations.

Bonus

Implement the following operations as an additional exercise.

filter - Takes a *predicate*(function, lambda or anonymous class) as an argument and returns a list only containing the elements which satisfy that *predicate*.
map - Takes a function as an argument and returns a list obtained by applying the function it to each of the items in the list.

Tips:

1. Use recursion instead of iteration.
2. Do not reassign any variable.
3. Do not mutate the original list in your functions.
4. Don’t worry about memory consumption.