**The List Abstract Data Type**

Recall that if *m* and *n* are integers, then {*m*..*n*} is the set of integers that are greater than or equal to *m* and less than or equal to *n.*

For any class *T*, there is a class *List*(*T*) with the following properties:

1. *len* is a function from *list*(*T*) to natural numbers, known as the *length* function.
2. If *n* is a natural number and *xi*∈ **N** for each *i* in {0..0-1}, then there exists a list(T) L of length *n* such that L[*i*]=*xi* for i in {0.*.n*-1}; we will write [*x*0,...,*x*n-1] for this list.
3. If *L* ∈ *List*(*T*) and *i* ∈ {0..*len*(*L*)-1}, then *Li* ∈ *T*.
4. If *L, M* ∈ *List*(*T*), then *L*=*M* if and only if (1) *len*(*L*) = *len*(*M*), and (2) *Li* = *Mi* for each *i* in {0..*len*(*L*)-1}.

The operation listed in Axiom 3 is called the *list indexing operator* or *list subscripting operator.* Alternative notations for *L.i* used in the literature (and in programming languages) include *L*[*i*], and *L*(*i*). In plain text we may type *L*[*i*].

Let *x* vary over *T,* and let *L*, *M, N, N'* vary over *list*(*T*). The following operations are defined on lists.

1. *x* is a***member*** *of L* if there is an *i* in {0..*len*(*L*)-1} such that *Li* = *x*.

For example,

* 20 is a member of the list of integers [10,20,30].
* NULL is a member of the list [NULL, NULL] (which is a list of pointers)
* The list [] has no members.
* [[],[1],[5,1]] is a list of lists of integers, and [1] is a member of it, but 1 is not a member of it.

1. The ***concatenation*** *of L and M*, written *L* + *M,* is the list of length *len*(*L*)+*len*(*M*) whose *i*'th member is *Li* for *i* in {0..*len*(*L*)-1}, and whose *i*'th member is *Mi*-*len*(*L*) for *i* in {*len*(*L*).. *len*(*L*)+*len*(*M*)-1}.

* [10,20] + [30,40,50] = [10,20,30,40,50]
* [10,20]+ [] = [10,20]
* []+[] = []
* [30,30,30]+[10,50] = [30,30,30,10,50]

1. An ***occurrence*** *of x in L* is a positive integer *i* such that *Li* = *x*.

For example,

* There are two occurrences of the number 5 in the list [5,3,4,5,1]
* There are no occurrences of the number 10 in the list [5,3,4,5,1]

1. An ***initial segment*** of *L* is a list *M* such that there exists a list *N* satisfying *M*+*N* = *L*.

For example,

* [1,2] is an initial segment of [1,2,3,4]
* the empty list [] is an initial segment of every list
* every list is an initial segment of itself.

1. If *L* is a nonempty list, the ***head*** *of L* is *L*0

For example,

* 1 is the head of [1,2,3]
* 3 is the head of [3,2,1]
* 10 is the head of [10]
* [1] is the head of [[1],[2],[3]]
* The empty list [] has no head.

1. If *L* is a nonempty list, the ***tail*** *of L* is the list *M* such that [*L*0] + *M* = *L* .

For example,

* [2,3] is the tail of [1,2,3]
* [] is the tail of [10]
* [[2],[3]] is the tail of [[1],[2],[3]]
* The empty list [] has no tail.

1. If ≤ is a total order on *T*, then *L* is ***sorted by ≤***  if *Li*≤ *Li*+1 for all *i* in {0..*len*(*L*)-2}

For example,

* [1,2,2,10] is sorted by the standard ordering ≤ on numbers
* ["bar","foo","football"] is sorted in lexigraphic order on strings
* the empty list [] is sorted (under any ordering)
* a list of length one is always sorted under any order.
* [1,2,0] is not sorted by the standard ordering on numbers
* [2,1,0] is sorted by >=.

1. A ***permutation of L*** is a list *M* such that, for each *x* in *T*, *x* occurs the same number of times in *L* and *M.*

For example,

* [1,2,3] is a permutation of [2,1,3]
* [1,2,3] is not a permutation of [1,2]
* [] is not a permutation of [1,2,3]
* [NULL,NULL,NULL] is a permutation of [NULL,NULL,NULL]
* [] is a permutation of []
* [1,1,2] is not a permutation of [1,2]
* Every list is a permutation of itself
* [[1,2],[3]]] is a permutation of [[3],[1,2]]

1. ***sort***(*L*,≤) is the permutation of *L* that is sorted by ≤.

For example, using the standard ordering on numbers,

* *sort*([40,20,10,10,40]) = [10,10,20,40,40]
* *sort*([]) = []

1. *L* is ***empty*** if *len*(*L*) = 0; otherwise, *L* is ***nonempty***.

For example,

* [] is empty
* [10,20] is nonempty
* [[]] is nonempty
* [NULL] is nonempty

1. A ***sublist of L*** is a list *M* such that there exist lists *N* and *K* satisfying *N*+*M*+*K* = *L*.

For example,

* [10,20] is a sublist of [10,20,30]
* [] is a sublist of every list.
* [10,30] is not a sublist of [10,20,30]

1. A ***subsequence of L*** is a list *M* such that there is an increasing (but possibly empty) sequence of natural numbers *k*0 < *k*1 < ... < *klen*(*M*)-1 with *Mi* = *Lk(i)* for each *i* in {0..*len*(*M*)}.

For example,

* [1,3,5] is a subsequence of [1,2,3,4,5]
* [10,30] is a subsequence of [10,20,30]
* [10,20,20] is not a subsequence of [10,20]
* [20,10] is not a subsequence of [10,20]
* The empty list is a subsequence of every list.
* Every sublist of *L* is a subsequence of *L*.

1. If *i* and *j* are natural numbers and 0 <= *i* <= *j* <= *len*(*L*), then *Li*:*j* is the list of length *j*-*i*, whos*e n'*th member is L*i+n*, for each *n* in {0..*j-i-*1}.

For example, if *L* = [10,20,30,40], then

* *L*0:1 = [10]
* *L*1:3 = [20,30]
* *L*2:2 = []
* *L*0:4 = *L*

1. We may consider members of a list either *distinctly* or *with multiplicity.* When looking at members of a list distinctly, we do not count multiple occurrences. When looking at members of a list with multiplicity, we count each occurrence separately.

For example,

* The list [1,2,2,4,1] has three distinct members, but five members with multiplicity.
* The sum of the distinct members of [1,2,3,3] is 6, but the sum of its members with multiplicity is 9.

**Exercises:**

Let *L* = [9,0,19,3,3]

1. How many distinct members does *L* have?
2. What is the sum of the member of *L*, with multiplicity?
3. what is *sort*(*L*)?
4. what is the tail of *L*? What is the head of *L*?
5. If *L* is a list with head *H* and tail *T*, is it true that *L* = *H*+*T*?
6. What is the length of the list [[9,4,0,3,20],[39,-3,0]] ? What type of object is it a list of?
7. Can a list have infinite length?
8. If *M* is a list, does every permutation of *M* have to be of the same length as *M*?
9. If a list has length 3, how many permutations does it have?
10. Can two different permutations of a list both be sorted? Can two different permutations of a list both be sorted by the same ordering?
11. how many permutations does *L* have?
12. How many sublists does *L* have?
13. Is the list [10, 9, 8] sorted? Is the list [10, 15, 10] sorted?
14. How many subsequences does *L* have?
15. Can the head of a list be a list?
16. Can the tail of a list be a number?
17. If *M* is a list, can the tail of *M* be equal to *M*?
18. if *M* is a list, can *M*+*M* = *M* ?
19. what is *L*[1:2] ?
20. implement vector<int> cat(vector<int> A, vector<int> B), which returns the concatenation of *A* and *B*.
21. implement vector<int> sort(vector<int> A), which returns *sort*(*A*)
22. implement vector<int> slice(vector<int> A, int i, int j), which returns *A*[*i*:*j*] if it is defined.
23. implement bool sublist(vector<int> A,vector<int> B), which is true if *A* is a sublist of *B* and false otherwise.
24. implement bool subseq(vector<int> A, vector<int> B), which is true if *A* is a subsequence of *B*, and false otherwise.
25. implement vector<vector<int>> subseqs(vector<int> A), which returns a vector of all of the subsequences of *A,* each occurring only once.
26. implement vector<vector<int>> sublists(vector<int> *A*), which returns a vector of all sublists of *A,* each occurring only once.
27. implement bool isPerm(vector<int> A, vector<int> B), which returns true if *A* is a permutation of *B* and false otherwise.
28. implement vector<vector<int>> allPerms(vector<int> *A*), which returns a vector of all of the permutations of *A*, each occurring only once.