A few more examples by recursion.
Per nutations (bijective functions (1-Londo)
Permutations (bijective fanctions (1-bonto)
from a set to it self.
Another way to think of it is all
Another way to think of it is all re-orderings of a last.
E_{\times} , S_{wy} list = 123.
Then pernutations of 123 we
[2] 3 Realist of Size n,
1 3 2 Re n list of Size n, there are n! pernultions
3 1 2
3 2
32
I dea: for every elevent, give it a town to
I dea: for every elevent, give it a town to be last, and paralle the first n-1
I dea: for every elevent, give it a town to be last, and paralle the first n-1 elements in all possible ways.
I dea: for every elevent, give it a town to be last, and paralle the first n-1 elements in all possible ways.
I dea: for every elenent, give it a town to be last, and permite the first n-1 elements in all possible ways. All permittings
I den: For every elevent, give it a town to be last, and permite the first n-1 elements in all possible ways.
I have ading of 123

```
Say N=1 (N=Size of list)
     Base case?
    L= [1]
    Then Perns (L) = [[13].
Sketch in Ctt: (N = L.5;2e())
vector < vector < int > > perms (vector < int > L)
      if (L. size () == 1) {
         Vector ( redor ( Int ) ) P;
           P. Push-back (L);
         J retour P;
    int N = L. Size(); // rot. val.
     for (int i = 0; i < L. size () itt) {
         1) put LC:3 last:
        Swap (L[i], L[n-i]);
        int x = L [ n-1];
       L. pop-back ();
      vector (vector (int)) T = perns (L);
       for (j=0; j(T-SiZe(); j++) {
            T [i], push_back (x);
             P. pash-back (TC; 3);
      L. Push buch (X);
swap (LCi3, LCn-13);
```

