Partitions

This problem is about representing partitions of the set $\{0, 1, 2, \ldots, n-1\}$. A partition of a set is a collection of disjoint non-empty subsets of the set, whose union is the whole set. In other words, every element in the set belongs to exactly one of the subsets, and each subset is non-empty. Such a partition can be represented uniquely by what is called a restricted growth sequence $p_0, p_1, \ldots, p_{n-1}$, that satisfies $p_0 = 0$ and $0 \leq p_i \leq 1 + \max_{0 \leq j < i} p_j$ for $i > 0$. Here, p_i denotes the 'part number' of the subset containing i. The parts are numbered in increasing order of their smallest elements. The part containing 0 is always 0. If element i is in the same part as some smaller element j, $p_i = p_j$, but if it is the smallest element in its part then i will be in a new part with number $1 + \max_{0 \leq j < i} p_j$. Thus the partition $\{\{0, 3\}, \{1, 2\}\}$ would be represented by the sequence $0, 1, 1, 0$. Given this representation, we can define an ordering on the partitions based on the lexicographic ordering of the sequences. A partition $p_0, p_1, \ldots, p_{n-1}$ is less than $q_0, \ldots, q_{n-1}$ if there exists an index i such that $p_i < q_i$ and $p_j = q_j$ for all $0 \leq j < i$. The rank of a partition is the number of partitions that are less than it. The partition $0, 0, \ldots, 0$ will have rank 0, and the partition $0, 1, 2, \ldots, n-1$ will have the highest rank. The rank of $0, 1, 1, 0$ is 8.

You have to write a function that will take a restricted growth sequence as input, and output its rank. You have to also write a function that will take the rank and output the corresponding sequence. Such functions are useful because we can work with just numbers instead of partitions.

Input/Output. The first line of input will specify n and the number of test cases t. The next t lines will contain one test case each. Each test case will be of the form R p_0 p_1 ... p_{n-1} or U r. Here $p_0, p_1, \ldots, p_{n-1}$ is a restricted growth sequence whose rank is to be computed, and r is the rank for which the corresponding sequence is to be found. R and U indicate the type of operation to be performed. n will be at most 20 and t will be at most 10^6.

The output should print out the answer for each test case in the same order as given, one case per line. For an R operation, a single number must be output, while for a U operation a sequence p_0 ... p_{n-1} must be output on a single line, the numbers separated by a space. Note that the ranks may be large so use long long int for calculations. Time required should not be more than few seconds. You can reduce the time required for I/O by setting the flag `ios_base::sync_with_stdio(false)`.

Hint. To know how many sequences are less than a given sequence, we need to know for any given partial sequence, how many sequences of length n start with that sequence. For example, if a sequence starts with $0, 1$ any sequence starting with $0, 0$ will be less than it. If we can count this easily, we can find the rank. The crucial observation is that the number of sequences starting with a given sequence depends only on the length and maximum value of the sequence. If we can compute this number for each possible value of the maximum and length and store it, we can use it whenever needed, without recomputing it. This is a technique called dynamic programming that is used heavily in designing efficient alogorithms. Can you compute these numbers easily using induction?

Submission. Submit a single file named RollNo_3.cpp .