Upwards

Filename: upwards

In a previous contest hosted by Kyle, the question was posed whether or not an input word was an “upword.” In order to be such a word, all of its letters have to appear in alphabetical order, with no repeated letters. For example, “act” is an “upword”, but “cat” and “deep” are not. Arup really likes these words and wants you to investigate them further. In addition, he’s added a new definition. The original “upward” is a level-0 upward. In order to be a level-1 upward, all the letters in the word have to be in alphabetical order and the gap between consecutive letters must contain at least one letter. Thus, “ace” is a level-1 upward, but “hit” is not. In general, we can define a level-k upward to be a word in alphabetical order where the gap between consecutive letters must contain at least k letters.

Now that Arup has made his definition, he wants to know of all the level-k upwards of exactly n letters, which one is of a particular rank, r, given values for k, n and r. The rank of a word in the list is based on alphabetical order. For example, of all level-1 upwords of length 3, “ace” has rank 1 (it’s the first) and “act” has rank 16.

The Problem

Given a level, k, number of letters, n, and a positive integer rank, r, find the r^{th} word in alphabetical order of all the level-k upwords of length n.

The Input

The first line of the input file will contain a single positive integer, c $(c \leq 100)$, representing the number of input cases. The input cases follow, one per line. Each of these lines will have the integers, k $(0 \leq k \leq 24)$, n $(2 \leq n \leq 26)$ and r $(1 \leq r \leq 10000)$, respectively, separated by spaces. Note that r is bounded by 10000, so even if a very large number of words exist, no more than 10000 will have to be generated. Also, r is guaranteed to be less than or equal to the total number of k-level upwords with exactly n letters.

The Output

For each case, output the correct string for the query, in all lowercase letters.

Sample Input

```
2
1 3 16
0 25 24
```

Sample Output

```
act
abdefghijklmnopqrstuvwxyz
```