I-Keyboard

Most of you have probably tried to type an SMS message on the keypad of a cellular phone. It is sometimes very annoying to write longer messages, because one key must be usually pressed several times to produce a single letter. It is due to a low number of keys on the keypad. Typical phone has twelve keys only (and maybe some other control keys that are not used for typing). Moreover, only eight keys are used for typing 26 letters of an English alphabet. The standard assignment of letters on the keypad is shown in the left picture:

1	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz
*	0 *space*	#

1	2 abcd	3 efg
4 hijk	5 lm	6 nopq
7 rs	8 tuv	9 wxyz
*	0 *space*	#

There are 3 or 4 letters assigned to each key. If you want the first letter of any group, you press that key once. If you want the second letter, you have to press the key twice. For other letters, the key must be pressed three or four times. The authors of the keyboard did not try to optimise the layout for minimal number of keystrokes. Instead, they preferred the even distribution of letters among the keys. Unfortunately, some letters are more frequent than others. Some of these frequent letters are placed on the third or even fourth place on the standard keyboard. For example, S is a very common letter in an English alphabet, and we need four keystrokes to type it. If the assignment of characters was like in the right picture, the keyboard would be much more comfortable for typing average English texts.

ACM have decided to put an optimised version of the keyboard on its new cellular phone. Now they need a computer program that will find an optimal layout for the given letter frequency. We need to preserve alphabetical order of letters, because the user would be confused if the letters were mixed. But we can assign any number of letters to a single key.

Input

There is a single positive integer T on the first line of input (equal to about 2000). It stands for the number of test cases to follow. Each test case begins with a line containing two integers K, L ($1 <= K <= L <= 90$) separated by a single space. K is the number of keys, L is the number of letters to be mapped onto those keys. Then there are two lines. The first one contains exactly K characters each representing a name of one key. The second line contains exactly L characters representing names of letters of an alphabet. Keys and letters are represented by digits, letters (which are case-sensitive), or any punctuation characters (ASCII code between 33 and 126 inclusively). No two keys have the same character, no two letters are the same. However, the name of a letter can be used also as a name for a key.

After those two lines, there are exactly L lines each containing exactly one positive integer F_1, F_2,

... F_L. These numbers determine the frequency of every letter, starting with the first one and continuing with the others sequentially. The higher number means the more common letter. No frequency will be higher than 100000.

Output

Find an optimal keyboard for each test case. Optimal keyboard is such that has the lowest "price" for typing average text. The *price* is determined as the sum of the prices of each letter. The price of a letter is a product of the letter frequency (F_i) and its position on the key. The order of letters cannot be changed, they must be grouped in the given order.

If there are more solutions with the same price, we will try to maximise the number of letters assigned to the last key, then to the one before the last one etc.

More formally, you are to find a sequence P_1, P_2, ... P_L representing the position of every letter on a particular key. The sequence must meet following conditions:

- $P_1 = 1$
- for each $i>1$, either $P_i = P_{i-1}+1$ or $P_i = 1$
- there are at most K numbers P_i such that $P_i = 1$
- the sum of products S_P = Sum[i=1..l] $F_i.P_i$ is minimal
- for any other sequence Q meeting these criteria and with the same sum $S_Q = S_P$, there exists such M, $1 <= M <= L$ that for any J, $M<J <= L$, $P_J = Q_J$, and $P_M>Q_M$.

The output for every test case must start with a single line saying Keypad #*I*:, where *I* is a sequential order of the test case, starting with 1. Then there must be exactly *K* lines, each representing one letter, in the same order that was used in input. Each line must contain the character representing the key, a colon, one space and a list of letters assigned to that particular key. Letters are not separated from each other.

Print one blank line after each test case, including the last one.

Example

Sample Input:

```
1
8 26
23456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
3371
589
1575
1614
6212
971
773
1904
2989
123
209
1588
```

1513
2996
3269
1080
121
2726
3083
4368
1334
518
752
427
733
871

Sample Output:

Keypad #1:
2: ABCD
3: EFG
4: HIJK
5: LM
6: NOPQ
7: RS
8: TUV
9: WXYZ

Warning: large Input/Output data, be careful with certain languages