C++ enables several functions of the same name to be defined, as long as these functions have different sets of parameters (at least as far as the parameter types or the number of parameters or the order of the parameter types are concerned). This capability is called function overloading. When an overloaded function is called, the C++ compiler selects the proper function by examining the number, types and order of the arguments in the call. Function overloading is commonly used to create several functions of the same name that perform similar tasks.
Your goal is to write several overloaded functions to output a character matrix.
Requirement: Use the sample main function and implement overloaded functions in the following two separate files "overloaded_functions.h" and "overloaded_functions.cpp". Use C++-style input/output.
Prohibited: Use multiple functions of different names. Use C-style input/output.

Input

The first line contains an integer N, representing the number of cases. In the following N lines, each line contains an integer n, representing the number of parameters to be inputted, followed by n integers which in turn correspond to (if exist) the first character c_1, the row number row, the column number col, the second character c_2, and the third character c_3. The input ends with a single zero.

Output

For each line of input, output the character array according to the number of parameters:
n=1: Output the character c_1.
n=2: Output the row-by-one matrix with each element equals to c_1.
n=3: Output the row-by-col matrix with each element equals to c_1.
n=4: Output the row-by-col matrix with each diagonal element equals to c_1, and each off-diagonal element equals to c_2.
n=5: Output the row-by-col matrix with each diagonal element equals to c_1, and each element in the upper triangular (echelon) equals to c_2, and all other elements equal to c_3.
The matrix is formed by separating adjacent columns with a space, and placing each row in a separate line.
Each output matrix should be separated by a newline.

Sample Input

```
5
1 C
2 A 2
3 B 3 2
4 D 3 3 A
5 E 4 5 C B
```

Sample Output

```
C

A
A

B B
B B
B B

D A A
A D A
A A D

E C C C C
B E C C C
B B E C C
B B B E C
```