ACM-ICPC Live Archive

4403 ASCII Diamond

ASCII diamonds can be drawn with integer side lengths. Each layer of this diamond is drawn with a single ASCII alphabet, starting with 'a' and ending with 'z' (starting from the center) and continues in cyclic order.

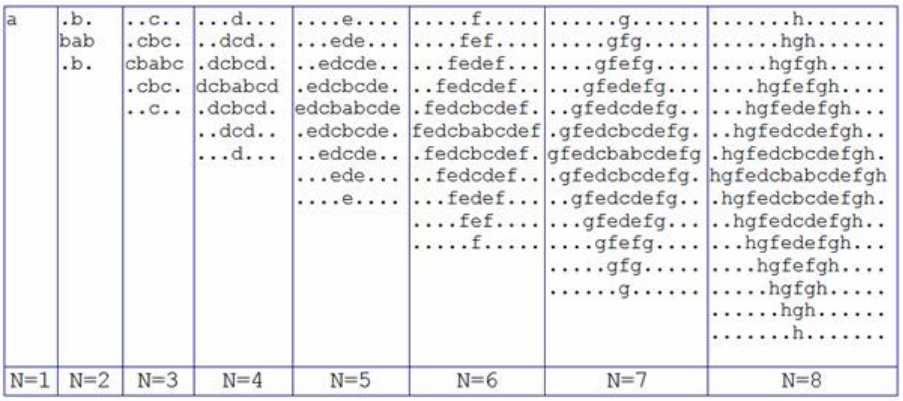
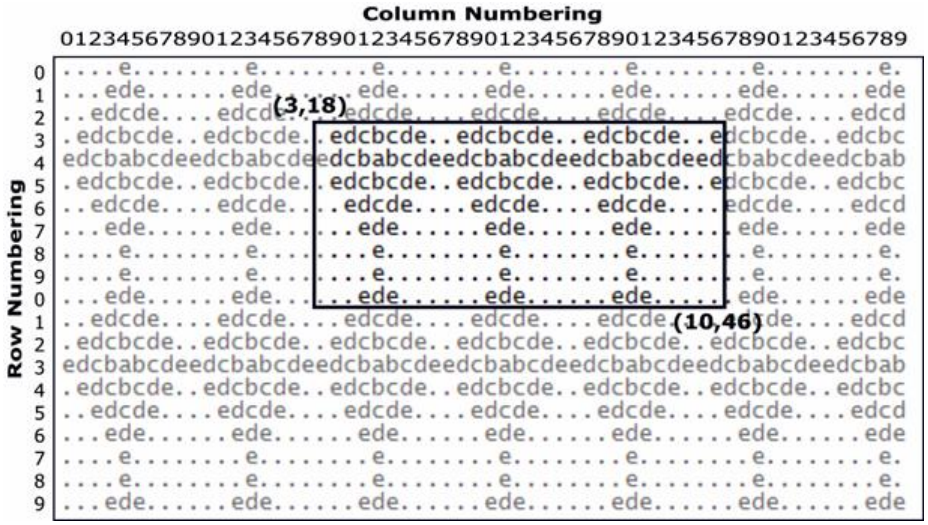


Figure 1: ASCII diamond for different side lengths.

Any one of these **ASCII** diamonds can be used to draw an infinite plane by using this as a tile. For example **ASCII** diamond of length 5 can be used to draw such an infinite grid. Only first 20 row and 60 columns are shown below:



Here rows and columns are numbered starting from zero. By specifying the topmost row (row_1), leftmost column (col_1), bottommost row (row_2) and rightmost column (col_2) we can specify a portion of such an infinite grid (also shown in figure above).

Given the side length of the tile to be used, the topmost row (row_1), leftmost column (col_1), bottommost row (row_2) and rightmost column (col_2) you have to print the pattern within these four boundaries (inclusive).

Input

Input contains at most 125 sets of inputs. But not all cases are extreme.

Each set of input contains five integers: N ($0 < N \leq 20000$), row_1, col_1, row_2, col_2 ($0 \leq row_1 \leq row_2 \leq 20000$, $0 \leq col_1 \leq col_2 \leq 20000$, $0 \leq (row_2 - row_1 + 1) * (col_2 - col_1 + 1) \leq 40000$). Here M denotes that the side length of the tiles used to draw the plane should be N. The meaning of row_1, col_1, row_2, col_2 are given in the problem statement.

The first sample input corresponds to the figure above.

Input is terminated by a line where the first integer is zero.

Output

For each line of input produce $(row_2 - row_1 + 2)$ lines of output. First line contains serial of output. Each of the next lines contain $(col_2 - col_1 + 1)$ characters. These lines describe the patterns within the specified rows and columns. Look at the output for sample input for details. The output file size is less than 1 MB.

Sample Input

```
5 3 18 10 46
100 50 50 69 69
0 2 3 4 5
```

Sample Output

```
Case 1:
.edcbcde..edcbcde..edcbcde..e
edcbabcdeedcbabcdeedcbabcdeed
.edcbcde..edcbcde..edcbcde..e
..edcde....edcde....edcde....
...ede......ede......ede.....
....e........e........e......
....e........e........e......
...ede......ede......ede.....
Case 2:
utsrqponmlkjihgfedcb
tsrqponmlkjihgfedcba
srqponmlkjihgfedcbaz
rqponmlkjihgfedcbazy
qponmlkjihgfedcbazyx
ponmlkjihgfedcbazyxw
onmlkjihgfedcbazyxwv
nmlkjihgfedcbazyxwvu
mlkjihgfedcbazyxwvut
lkjihgfedcbazyxwvuts
kjihgfedcbazyxwvutsr
jihgfedcbazyxwvutsrq
ihgfedcbazyxwvutsrqp
hgfedcbazyxwvutsrqpo
gfedcbazyxwvutsrqpon
fedcbazyxwvutsrqponm
edcbazyxwvutsrqponml
```

```
dcbazyxwvutsrqponmlk
cbazyxwvutsrqponmlkj
bazyxwvutsrqponmlkji
```