Selection Contest for HKU ACM Team 2018

The University of Hong Kong

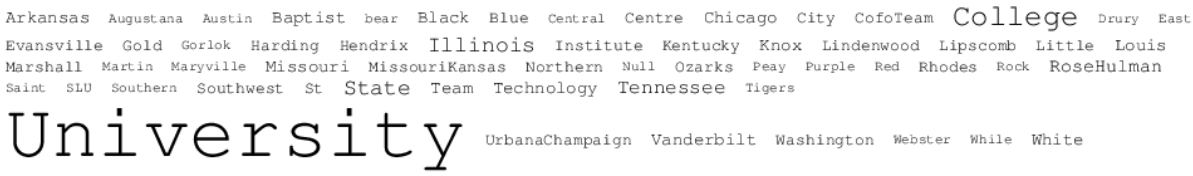
September 7, 2018

Important Notes

- The contest begins at **7:00 pm** and lasts for **150 minutes**, ending at **9:30 pm**.
- The problem set consists of **5 problems**. The problems may not be ordered by difficulty; remember to read all the problems.
- Printed and written notes are allowed, while internet access is prohibited.
- Please use **standard input/output** (e.g. `scanf`/`printf`, `cin`/`cout` or `System.in`/`System.out.print`) and do not print anything other than those required in the problem.
- For each problem, there are no more than **40 test cases** and the running time limit is **4 seconds**.
- Additional note on Problem A (6816): a single line with a pair of space-separated zeros indicates the end of input.
- Additional note on Problem B (6876): please output the test case numbers ("`Case x:`"), as shown in Sample Output.

6816 Word Cloud

A word cloud (or tag cloud) is a visual representation of textual data based on a weighted metric. In the cloud below (which is based on this year's list of Mid-Central teams), the font size of each word is based on its number of occurrences in the data set. Tagg Johnson is a man obsessed with counting words that appear in online documents. On his computer, he keeps a spreadsheet of all the sites he visits, along with a list of words that appear on each site and the number of times such word appears. Tagg would like to generate word clouds based on the data he has collected.



Before describing the algorithm Tagg uses for generating clouds, we digress for a quick lesson in typography. The basic unit of measure is known as a *point* (typically abbreviated as *pt*). A font's size is described based on the vertical number of points from one line to the next, including any interline spacing. For example, with a 12pt font, the vertical space from the top of one character to the top of a character below it is 12 points. We assume that a character's height is precisely equal to the font's point size (regardless of whether the character is upper or lower case).

For this problem, we focus on a fixed-width font, such as Courier, in which each character of the alphabet is also given the same amount of width. The character width for such a font depends on the font size and the aspect ratio. For Courier, a word with t characters rendered in a font of size P has a total width of $\left\lceil \frac{9}{16} \cdot t \cdot P \right\rceil$ when measured in points. Note well the use of the ceiling operator, which converts any noninteger to the next highest integer. For example, a 5-letter word in a 20pt font would be rendered with a height of 20 points and a width equal to $\left\lceil \frac{900}{16} \right\rceil = \lceil 56.25 \rceil = 57$ points.

Now we can describe Tagg's algorithm for creating a word cloud. He pre-sorts his word list into alphabetical order and removes words that do not occur at least five times. For each word w, he computes a point size based on the formula $P = 8 + \left\lceil \frac{40(c_w - 4)}{(c_{max} - 4)} \right\rceil$, where c_w is the number of occurrences of the word, and c_{max} is the number of occurrences of the most frequent word in the data set. Note that by this formula, every word will be rendered with anywhere from a 9pt font to a 48pt font. He then places the words in rows, with a 10pt horizontal space between adjacent words, placing as many words as fit in the row, subject to a maximum width W for his entire cloud. The height of a given row is equal to the *maximum* font size of any word rendered in that row.

As a tangible example, consider the following data set and word cloud.

word	count
apple	10
banana	5
grape	20
kiwi	18
orange	12
strawberry	10

In this example, `apple` is rendered with 23pt font using width 65pt, `banana` is rendered with 11pt font using width 38pt, and `grape` is rendered with 48pt font and width 135pt. If the overall word cloud is constrained to have width at most 260, those three words fit in a row and the overall height of that row is 48pt (due to `grape`). On the second row `kiwi` is rendered with height 43pt and width 97pt, and `orange` is rendered with height 28pt and width 95pt. A third row has `strawberry` with height 23pt and width 130pt. The overall height of this word cloud is 114pt.

Input

Each data set begins with a line containing two integers: W and N. The value W denotes the maximum width of the cloud; $W \leq 5000$ will be at least as wide as any word at its desired font size. The value $1 \leq N \leq 100$ denotes the number of words that appear in the cloud. Following the first line are N additional lines, each having a string S that is the word (with no whitespace), and an integer C that is a count of the number of occurrences of that word in the original data set, with $5 \leq C \leq 1000$. Words will be given in the same order that they are to be displayed within the cloud.

Output

For each data set, output the word '`CLOUD`' followed by a space, a serial number indicating the data set, a colon, another space, and the integer height of the cloud, measured in font points.

Sample Input

```
260 6
apple  10
banana 5
grape  20
kiwi   18
orange 12
strawberry 10
250 6
apple  10
banana 5
grape  20
kiwi   18
orange 12
strawberry 10
610 6
apple  10
banana 5
grape  20
kiwi   18
orange 12
strawberry 10
0 0
```

Sample Output

```
CLOUD 1: 114
CLOUD 2: 99
CLOUD 3: 48
```

6876 A Cure for the Common Code

You've been tasked with relaying coded messages to your fellow resistance fighters. Each coded message is a sequence of lower-case letters that you furtively scrawl on monuments in the dead of night.

Since you're writing these messages by hand, the longer the message, the greater the likelihood of being caught by the evil empire while writing. Because of this you decide it would be worthwhile to come up with a simple encoding that might allow for shorter messages. After thinking about it for a while, you decide to use integers and parentheses to indicate repetition of substrings when doing so shortens the number of characters you need to write. For example, the 10 character string '`abcbcbcbca`' could be more briefly written as the 7 character string '`a4(bc)a`'

If a single letter is being repeated, parentheses are not needed. Also, repetitions may themselves be repeated, so you can write the 20 character string '`abbbcdcdcdabbbcdcdcd`' as the 11 character string '`2(a3b3(cd))`' and so forth.

Input

Each test case consists of a single line containing a string of lower-case letters of length ≤ 500. A line containing a single '`0`' will terminate the input.

Output

For each test case, output the number of characters needed for a minimal encoding of the string.

Sample Input

```
abcbcbcbca
abbbcdcdcdabbbcdcdcd
0
```

Sample Output

```
Case 1: 7
Case 2: 11
```

6818 Reverse Rot

A very simplistic scheme, which was used at one time to encode information, is to rotate the characters within an alphabet and rewrite them. ROT13 is the variant in which the characters A-Z are rotated 13 places, and it was a commonly used insecure scheme that attempted to "hide" data in many applications from the late 1990's and into the early 2000's.

It has been decided by Insecure Inc. to develop a product that "improves" upon this scheme by first reversing the entire string and then rotating it. As an example, if we apply this scheme to string 'ABCD' with a reversal and rotation of 1, after the reversal we would have 'DCBA' and then after rotating that by 1 position we have the result 'EDCB'.

Your task is to implement this encoding scheme for strings that contain only capital letters, underscores, and periods. Rotations are to be performed using the alphabet order:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ_.
```

Note that underscore follows Z, and the period follows the underscore. Thus a forward rotation of 1 means A is shifted to B, that is, 'A' → 'B', 'B' → 'C', ..., 'Z' → '_', '_'→ '.', and '.' → 'A'. Likewise a rotation of 3 means 'A' → 'D', 'B'→ 'E', ..., '.' → 'C'.

Input

Each input line will consist of an integer N, followed by a string. N is the amount of forward rotation, such that $1 \leq N \leq 27$. The string is the message to be encrypted, and will consist of 1 to 40 characters, using only capital letters, underscores, and periods. The end of the input will be denoted by a final line with only the number '0'.

Output

For each test case, display the "encrypted" message that results after being reversed and then shifted.

Sample Input

```
1 ABCD
3 YO_THERE.
1 .DOT
14 ROAD
9 SHIFTING_AND_ROTATING_IS_NOT_ENCRYPTING
2 STRING_TO_BE_CONVERTED
1 SNQZDRQDUDQ
0
```

Sample Output

```
EDCB
CHUHKWBR.
UPEA
ROAD
PWRAYF_LWNHAXWH.RHPWRAJAX_HMWJHPWRAORQ.
FGVTGXPQEAGDAQVAIPKTVU
REVERSE_ROT
```

6819 Shrine Maintenance

A religious sect has holy sites with shrines placed around a circle of radius 1000. The circle is split into N equal length arcs and the endpoints are numbered in order, 1 through N. The first figure shows a circle where N is 12, with 12 gray tick marks like on a 12-hour analog clock. We can imagine the marks numbered, as on a clock, with 12 at the top. Each circle has one or more *sacred numbers* associated with it. The sacred numbers for the circle in the first figure are 2 and 3. A shrine, indicated by a black dot in the figure, is placed at each mark whose number is a multiple of at least one of the sacred numbers, so in this case the shrines are at positions 2, 3, 4, 6, 8, 9, 10, and 12.

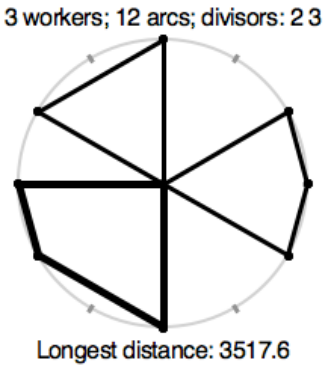


Figure 1

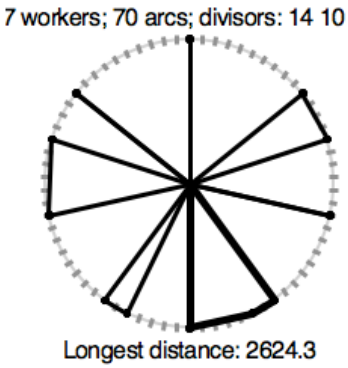


Figure 2

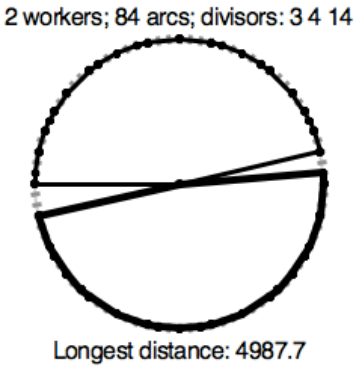


Figure 3

When it comes time to inspect and repair the shrines at a given site, the area is closed and a team of workers simultaneously fan out from a maintenance shed, located in the center of the circle, so that each shrine is visited by at least one worker. Once all workers have returned to the shed, the site is reopened to the public. Because these sites are in great demand, it is important that they be closed as briefly as possible. In order to minimize this time, they must figure out how to apportion the shrines among the current number of workers, so the maximum distance traveled by any one worker is as small as possible. Figure 1 shows one choice for the optimal solution paths for 3 workers. The lower left path has darker lines, to indicate that it is one with the longest length, which in this case is approximately 3517.6.

This sect has many circular sites with multiple shrines. The number of available workers at a site, W, the value of the number equal arcs, N, and the sacred numbers vary between sites. The sacred numbers are always divisors of N. Your job is to help figure out how much time is required for maintenance. Figures 2 and 3 show optimal solutions for other sites.

Input

The input consists of one or more data sets. Each data set is on a single line and consists entirely of positive integers. The first three entries are W, the number of workers, N, the number of equal arcs around the circle, and D, the number of sacred divisors of N. At the end come the D divisors of N. W is no more than the total number of shrines; $N \leq 8600$, and $D \leq 6$; each listed divisor of N is smaller than N.

A single zero, '0', will be placed on the last line to indicate the end of the input.

Output

The output is a single line for each dataset: the maximum distance a worker must travel with an optimal assignment of the shrines. This number is displayed so that it is rounded to one decimal place, and always shows that decimal place, even if it is 0. To ensure unique answers with double arithmetic, the datasets are chosen so that if your answer is anywhere within .005 of the exact minimum distance, then the answer rounded to one decimal place will be the same.

The first three sample datasets correspond to the three figures.

Caution: Be careful with your algorithm so it finishes rapidly.

Sample Input

```
3 12 2 2 3
7 70 3 14 10 35
2 84 3 3 4 14
4 35 2 7 5
3 20 2 5 4
3 6 1 1
4 6 1 1
1 6 1 1
8600 8600 3 1 10 100
0
```

Sample Output

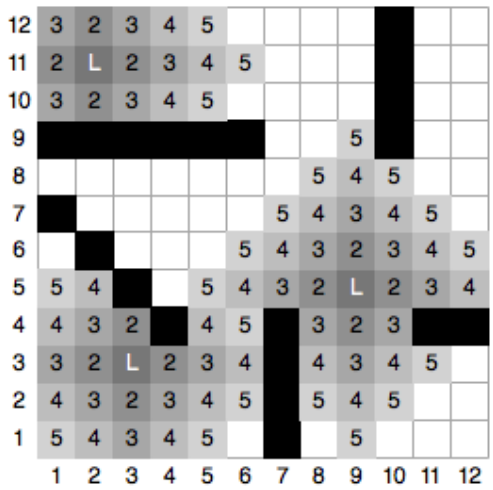
```
3517.6
2624.3
4987.7
3224.9
3488.4
3000.0
3000.0
7000.0
2000.0
```

6820 Wet Tiles

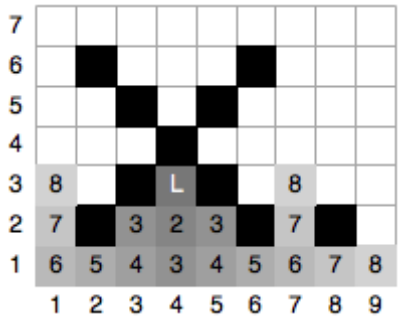
Alice owns a construction company in the town of Norainia, famous for its unusually dry weather. In fact, it only rains a few days per year there. Because of this phenomenon, many residents of Norainia neglect to do roof repairs until leaks occur and ruin their floors. Every year, Alice receives a deluge of calls from residents who need the leaks fixed and floor tiles replaced. While exquisite in appearance, Norainia floor tiles are not very water resistant; once a tile becomes wet, it is ruined and must be replaced. This year, Alice plans to handle the rainy days more efficiently than in past years. She will hire extra contractors to dispatch as soon as the calls come in, so hopefully all leaks can be repaired as soon as possible. For each house call, Alice needs a program to help her determine how many replacement tiles a contractor team will need to bring to complete the job.

For a given house, square floor tiles are arranged in a rectangular grid. Leaks originate from one or more known source locations above specific floor tiles. After the first minute, the tiles immediately below the leaks are ruined. After the second minute, water will have spread to any tile that shares an edge with a previously wet tile. This pattern of spreading water continues for each additional minute. However, the walls of a house restrict the water; if a damaged area hits a wall, the water does not penetrate the wall. We assume there are always four outer walls surrounding the entire house. A house may also have a number of additional "inner" walls; each inner wall is comprised of a connected linear sequence of locations (which may or may not be connected to the outer walls or to each other).

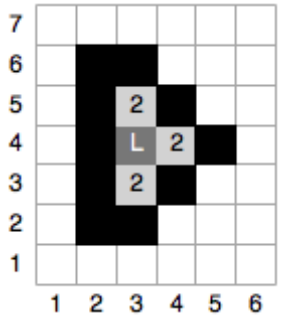
As an example, Figure 1 shows water damage (in gray) that would result from three initial leaks (each marked with a white letter 'L') after each of the first five minutes of time. Tiles labeled '2' become wet during the second minute, tiles labeled '3' become wet during the third minute, and so forth. The black areas designate inner walls that restrict the flow of water. Note that after 5 minutes, a total of 75 tiles have been damaged and will need to be replaced. Figures 2 through 4 show other houses that correspond to the example inputs for this problem.



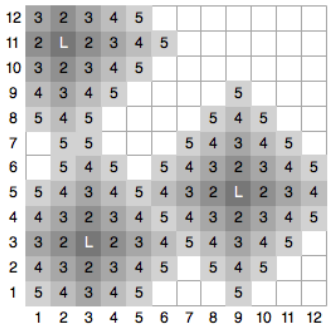
75 wet tiles
Figure 1



17 wet tiles
Figure 2



4 wet tiles
Figure 3



94 wet tiles
Figure 4

Input

Each house is described beginning with a line having five integral parameters: $X\ Y\ T\ L\ W$. Parameters X and Y designate the dimensions of the rectangular grid, with $1 \leq X \leq 1000$ and $1 \leq Y \leq 1000$. The coordinate system is one-indexed, as shown in the earlier figures. Parameter T designates the number of minutes that pass before a team of contractors arrives at a house and stops the leaks, with $1 \leq T \leq 200000$. The parameter L designates the number of leaks, with $1 \leq L \leq 100$. Parameter W designates the number of inner walls in the house, $0 \leq W \leq 100$.

The following $2L$ integers in the data set, on one or more lines, are distinct (xy) pairs that designate the locations of the L distinct leaks, such that $1 \leq x \leq X$ and $1 \leq y \leq Y$.

If $W > 0$, there will be $4W$ additional integers, on one or more lines, that describe the locations of the walls. For each such wall the four parameters (x_1, y_1), (x_2, y_2) describe the locations of two ends of the wall. Each wall replaces a linear sequence of adjoining tiles and is either axis-aligned or intersects both axes at a 45 degree angle. Diagonal walls are modeled as a sequence of cells that would just be touching corner to corner. If the two endpoints of a wall are the same, the wall just occupies the single cell at that location. Walls may intersect with each other, but no leak is over a wall.

There will be one or more houses in the data file and a line with a single integer '`-1`' designates the end of the data set.

Output

For each house, display the total number of tiles that are wet after T minutes.

Sample Input

```
12 12 5 3 5
2 11 3 3 9 5
1 9 6 9 1 7 4 4 7 1 7 4
10 9 10 12 11 4 12 4
9 7 8 1 3
4 3
2 2 6 6 6 2 2 6 8 2 8 2
6 7 50 1 3
3 4
2 2 2 6 3 6 5 4 5 4 3 2
12 12 5 3 0
2 11 3 3 9 5
```

```
-1
```

Sample Output

```
75
17
4
94
```