1. Apartments

Program Name: Apartments.java   Input File: apartments.dat

Mr. Jones is looking at buying one of several apartment complexes. Currently, most of the units in the complexes are rented but Mr. Jones is afraid that the tenant to unit ratio for a complex does not meet city code for density. You are to write a program that will help Mr. Jones determine if a complex's tenant to unit ratio meets the city code and the number of apartment units that are currently unoccupied. To meet the city density code, the tenant to unit ratio cannot exceed an average of 4 people per unit and includes only units that are occupied.

Input

The first line of input will contain a single integer `n` that indicates the number of apartment complexes that Mr. Jones is considering. For each apartment complex, the first line will contain an integer `m` which indicates the number of units in the apartment. Each of the following `m` lines will contain two integers separated by a space. The first integer is the apartment number and the second integer is the number of people currently living in that apartment.

Output

For each apartment complex, you will output `YES` if the complex meets the tenant to unit ratio or `NO` if it does not, a space, and the number of units that are empty.

Example Input File

```
2
8
3 5
2 4
7 0
5 3
8 3
9 5
10 3
14 0
10
101 5
102 4
103 5
104 3
105 5
201 0
202 2
203 5
204 6
205 3
```

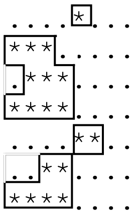
Example Output to Screen

```
YES 2
NO 1
```

2. Blob Top

Program Name: BlobTop.java Input File: blobtop.dat

James is studying different shapes in a plane. For this particular study, he refers to the shapes as “blobs” because they are irregularly shaped solid polygons. He represents his blobs in a rectangular grid as a collection of one or more contiguous asterisks (*). Contiguous means that the asterisks must be adjacent either horizontally or vertically. Characters in the grid that are not part of a blob are represented by periods (.). In the diagram below, there are 4 blobs.



```
....*...
***.....
.***....
****....
....**..
..**....
****....
```

You are to write a program that, will determine the location of the uppermost, leftmost character of a blob given the coordinates of a given character in the grid. The uppermost, leftmost character of the largest blob in the example above is row 2, column 1 or `2 1`.

Input

The first line of input will contain a single integer `n` that indicates the number of data sets to follow. For each data set:

- the first line will contain three integers in the form `r c s` which meet the following criteria:
  - `r` ≥ 3 is the number of rows in the grid
  - `c` ≥ 3 is the number of columns in the grid
  - `s` > 1 is the number of test cases for that grid
- the next `r` lines will contain the grid.
- the next `s` lines will each contain an ordered pair `x y`, 1 ≤ `x` ≤ `r` and 1 ≤ `y` ≤ `c`, which is the location of a character in the grid.

Output

For each test case, you will print the coordinates of the upper, leftmost character of the blob in the form `j k` where 1 ≤ `j` ≤ `r` and 1 ≤ `k` ≤ `c`. If the test case falls on a square that is not part of a blob, print `NOT A BLOB`,

Example Input File

```
2
7 8 2
....*...
***.....
.***....
****....
....**..
..**....
..**....
4 1
5 3
4 8 3
......**
***.****
......**
***.***.
2 3
4 5
2 8
```

2. Blob Top (cont.)

Example Output to Screen

```
2 1
NOT A BLOB
2 1
1 7
1 7
```

3. Buying a Car

Program Name: Buying.java   Input File: buying.dat

Roger does a lot of city driving and is very concerned about his car's gas mileage. In the average year, he drives about 12,000 miles. His current car is an old Jeep Cherokee, gets an average of 12 miles per gallon and he could sell it for $4,500. He is looking to buy one of several cars that receive better miles per gallon. He wants to look at his options and decide which purchase would be best for him.

Roger wants to know how much the new car would cost him after selling his old car and how much money he would save in gas over a two year period. Right now, the average price of gas is $2.65 per gallon and, for the purpose of this simulation, will remain constant over the next 2 years. You are to write a program that will make these comparisons for him.

Input

The first line of input will contain a single integer `n` that indicates the number of cars that Roger is researching. Each of the following `n` lines will contain the car name, the miles per gallon that the car averages and the price of the car. Each of items will be separated by two hyphens (`--`).

Output

For each car, you will print its name, a space, a dollar sign (`$`), the price in integer dollars he would have to pay for the new car after selling his old car, a space, a dollar sign (`$`), and the amount of money that he would save on gas over the 2 year period rounded to two decimal points.

Example Input File

```
3
Honda Element--19--17950
Ford Edge--19--18130
Toyota RAV4--20--18885
```

Example Output to Screen

```
Honda Element $13450 $1952.63
Ford Edge $13630 $1952.63
Toyota RAV4 $14385 $2120.00
```

4. Cancer Cells

Program Name: Cancer.java Input File: cancer.dat

While doing scientific research, scientists noticed a pattern to the growth and decay of a specific cancer. Even though the cells in the body are in three dimensions, for the purpose of this simulation we will consider a set of cells in a flat, square area. Additionally, cells are considered to be adjacent if they are contiguous either vertically, horizontally, or diagonally.

The scientists divided their observations into given intervals or time steps. After watching numerous time steps, they observed the following was true at the end of each time step:

- An infected cell that was adjacent to exactly two or exactly three infected cells at the beginning of the time step continued to be infected.
- An infected cell that was not adjacent to exactly two or exactly three infected cells at the beginning of the time step became uninfected.
- An uninfected cell at the beginning of the time step became infected if it was adjacent to exactly three infected cells.

Even though the cell containing the anti-cancer needle is not infected, for the purpose of the transition rules above, it is considered to be infected. In the example below, the **@** is the anti-cancer needle, the **#** is an infected cell, and the **.** is an uninfected cell. The minimum number of time steps in this example is 3.

Original	**Begin Time Step 1**	**End Time Step 1**	**Begin Time Step 2**	**End Time Step 2**	**Begin Time Step 3**	**End Time Step 3**
....@ ##... #.... ...#. ##.##	..... ##.@. #.... ...#. ##.##	..... ##.@. ###.. ##### ..###	..... ##@.. ###.. ##### ..###	.#... #.@.. ..... #...# ....#	.#... #..@. ..... #...# ....#	....@. ....

You are to simulate the growth and decay of a specific cancer in time steps as described above, where an anti-cancer needle has already been placed on one cell of a targeted area at the beginning of the simulation. For each time step:

- If possible, move the anti-cancer needle to an adjacent, uninfected cell. If not possible, the area cannot be freed of cancer.
- Mark each cell's future condition as infected or uninfected based on the application of the above rules to the cell's condition at the beginning of the time step.
- Make the noted changes to each cell to create the new state at the end of the time step.
- Repeat the time steps to trace the growth or decay of the cancer.

In the example above, the anti-cancer needle has been moved from its position in the "Original" box to a new position in the "Begin Time Step 1" box, but the state of the other cells remains unchanged from the "Original". Then the rules were applied to create the new state as shown in the "End Time Step 1". These steps are repeated to remove all the cancerous cells. The goal is to remove all the cancerous cells in as few time steps as possible.

Input

The first line of input will contain a single integer `n` that indicates the simulations to follow. For each simulation, the first line will contain a single integer `m`, $0 < m \leq 5$, that indicates the size of the square to be considered. The next `m` lines will each contain `m` characters as described above (no spaces).

Output

For each simulation, you will print a single integer indicating the minimum number of time steps required to free the area of cancer. If the area cannot be freed of cancer, print `-1`. If the area was initially free of cancer, print a 0.

Note: If the state of a given area does not change after five consecutive time steps, you may assume that the area cannot be freed of cancer.

4. Cancer Cells (cont.)

Example Input File

```
3
5
....@
##...
#....
...#.
##.##
3
.##
.#.
@##
3
##.
#..
@..
```

Example Output to Screen

```
3
10
-1
```

5. CDs

Program Name: CDs.java   Input File: cds.dat

Alexander is making some music CDs to use at some parties. He has a list of songs in the order that he wants to put on the CDs for each party. He also has the length of each song in minutes and seconds. You are to write a program for him to use to determine the number of CDs that he needs to buy for each party if each CD can hold no more than 20 minutes of music.

Input

The first line of input will contain a single integer `p` that indicates the number of parties he is creating CDs for. For each party, the first line will contain a single integer `n` that indicates the number of songs he has on his list for that party. Each the following `n` lines will contain two integers, `m` and `s`, that indicate the length of the song in minutes `m` and seconds `s`. The items will be separated by a space.

Output

For each party, you will output `PARTY #x: y`, where `x` is the party number and `y` is the number of CDs Alexander will need for that party.

Example Input File

```
2
18
5 25
4 58
4 45
5 35
4 56
5 23
4 34
3 45
5 34
5 45
7 23
5 34
4 45
4 0
5 3
13 34
4 56
5 52
10
5 56
6 12
4 56
5 35
4 45
7 23
5 56
5 43
4 9
5 0
```

Example Output to Screen

```
PARTY #1: 7
PARTY #2: 4
```

6. De-Expansion

Program Name: Deexpansion.java   Input File: deexpansion.dat

Something went horribly wrong with the encrypted document backup process at work and all files were converted to strings containing 1's and 0's representing what the binary representation of the file was! Your boss needs a program, fast, that can restore all the files. All the original files were in 16 bit Unicode.

Input

A file containing a long string of 1's and 0's whose length is guaranteed to be a multiple of 16.

Output

The original file contents.

Example Input File

```
00101010000000000001011000000000100101100000000011001110000000000000010000000
00010010110000000001100111000000000000001000000000000101110000000000001011000
00000010100110000000000000001000000000011110110000000000100111000000000001001011
000000000111001100000000010010110000000000111011000000000100001100000000000011
01100000000000000001000000000001100110000000001001011000000000000110110000000001
010011000000000011101000000000010110000000000000101000000000000001011000000000
00000101000000000000111000100000000011110110000000001111011000000000000100110000
00000000001000000000000101110000000000001011000000000100101100000000001110110
000000001110011000000000000001000000000010011110000000001111011000000000010101
11000000000000001000000000001001110000000001010011000000000110001100000000011
11011000000000011011100000000010100110000000000100111000000000101001100000000
00010011000000000000001000000000010010110000000000010111000000000100001000000
0000
```

(The actual input file contains only one line which appears to span many lines in the printed version.)

Example Output to Screen

```
This is the original file.

Good thing you recovered it!
```

7. Selection Sort

Program Name: Selection.java Input File: selection.dat

The algorithm for a standard selection sort is:

1. Find the minimum value in the list
2. Swap it with the value in the first position
3. Repeat the steps above for the remainder of the list (starting at the second position the second time through the list, and advancing the starting position each time through the list).

Dr. Martin wants to trace what is happening with his data when he uses this selection sort to sort a non-empty array of integers. You are to write a program for him that will print his data after each iteration of the selection sort.

Input

The first line of input will contain a single integer `n` that indicates the number of integer arrays to follow. Each of the following `n` lines will contain the integers contained in an array. The integers in a given array will be separated by a single space.

Output

For each array, you will print the state of the array on one line after each iteration of the standard selection sort, printing a space after each array element. Print a blank line after the last iteration through the array.

Example Input File

```
3
45 15 12 -5 14 -3 8
12 -23 43 56 43 -1 -5
8 -2 -4 0 17 3 6 9 2 -2
```

Example Output to Screen

```
-5 15 12 45 14 -3 8
-5 -3 12 45 14 15 8
-5 -3 8 45 14 15 12
-5 -3 8 12 14 15 45
-5 -3 8 12 14 15 45
-5 -3 8 12 14 15 45

-23 12 43 56 43 -1 -5
-23 -5 43 56 43 -1 12
-23 -5 -1 56 43 43 12
-23 -5 -1 12 43 43 56
-23 -5 -1 12 43 43 56
-23 -5 -1 12 43 43 56

-4 -2 8 0 17 3 6 9 2 -2
-4 -2 8 0 17 3 6 9 2 -2
-4 -2 -2 0 17 3 6 9 2 8
-4 -2 -2 0 17 3 6 9 2 8
-4 -2 -2 0 2 3 6 9 17 8
-4 -2 -2 0 2 3 6 9 17 8
-4 -2 -2 0 2 3 6 9 17 8
-4 -2 -2 0 2 3 6 8 17 9
-4 -2 -2 0 2 3 6 8 9 17
```

8. Smoothing an Image

Program Name: Smooth.java   Input File: smooth.dat

An image can be represented by a rectangular grid of pixels. Each pixel has associated with it a triplet of numbers giving the intensities of the red, green, and blue colors. These intensities are on a scale of 0 to 255. There are times that an image needs to be smoothed because of local imperfections in the image. The smoothing is done by replacing the offending pixel with the average value of the pixels surrounding that pixel.

This is a different version of the smoothing problem. The image will be a square matrix of numbers in the range 0 to 255. One intensity value will be associated with each pixel instead of three. The whole image will be smoothed instead of just particular regions in the image. The sub-grid over which the averaging is to be done will be specified in the problem. The sub-grid will be a square of odd dimension so that the pixel that is to be replaced will be at the center of the sub-grid. Obviously, the sub-grid will get truncated at the edges of the image. The average will include the value of the central pixel and is rounded to the closest integer value.

Input

The first line of input will contain two integers `n` and `m`:

- The integer `n` indicates the dimension of the square image.
- The integer `m` is odd and smaller than `n` and indicates the dimension of the sub-grid over which the average has to be taken .
- Each of the next `n` lines of data will contain `n` integer numbers in the range 0 to 255 separated by one or more spaces.

Output

You will print out the smoothed image. The smoothed image will be `n` lines of data each line having `n` integer values in the range 0 to 255 followed by a single space.

Example Input File

```
10 3
  65  223  255  133  221   95  141   41  172  127
 177   37   68    0  224  196  243  145   61   75
 236  151  207  197   41  106  120  216  215  159
 226   57  176   30  224   67  217  244  246   22
 226   57   27   31   46  101  250  255  234  160
 100  140  250  184   73  206   90  212  131    9
 109  147  116  226  217  238  117  244  187  198
  24   19   86  162    5  227  189    1   41   21
  30   49  169  238  149  158  112   87  206  211
 181  112   54  199  196  106  174   63    6   73
```

Example Output to Screen

```
126 138 119 150 145 187 144 134 104 109
148 158 141 150 135 154 145 150 135 135
147 148 103 130 121 160 173 190 154 130
159 151 104 109 94 130 175 222 195 173
134 140 106 116 107 142 182 209 168 134
130 130 131 130 147 149 190 191 181 153
90 110 148 147 171 151 169 135 116 98
63 83 135 152 180 157 153 132 133 144
69 80 121 140 160 146 124 98 79 93
93 99 137 168 174 149 117 108 108 124
```

9. Superhero Day

Program Name: Superhero.java Input File: superhero.dat

For spirit week, your school is having several special days for students to show their school spirit. On one of the days, students are encouraged to dress as their favorite superhero. You have decided to dress as Superman but you need a Superman logo for your shirt. You are to design one exactly like the one below so you can print it on an iron-on transfer. However, since it is a transfer, you will have to print a reverse or mirror image so it will read correctly when transferred to your shirt.

Input

The input file consists of the image of the Superman logo shown below.

Output

You will print the reverse image, shown below, that would be used as a transfer to your shirt.

Example Input File

```
    *****************
   ***$$$$$$$$$$$$$$**
  ** $$         $$ $ **
 ** $$           $$$  **
** $$$                 **
 *$$$$$$$$$           **
  *$$$$$$$$$$$$$$    **
   **$$$$$$$$$$$$$$$**
    **      $$$$$$$**
     **         $$**
      **$$$    $$**
       *$$$$$$$$**
        **     **
         **   **
          ** **
           ***
            *
```

Example Output to Screen

```
    *****************
   **$$$$$$$$$$$$$$***
  ** $ $$         $$ **
 **  $$$           $$ **
**                 $$$ **
 **           $$$$$$$$$*
  **    $$$$$$$$$$$$$$*
   **$$$$$$$$$$$$$$$**
    **$$$$$$$      **
     **$$         **
      **$$    $$$**
       **$$$$$$$$*
        **     **
         **   **
          ** **
           ***
            *
```

10. Unwrapping a Spiral

Program Name: Unwrap.java   Input File: unwrap.dat

Given a square matrix of integers, we can unwrap it by following a spiral path. The spiral path repeats the following sequence until it terminates:

Left to Right
Top to Bottom
Right to Left
Bottom toTop

Input

The first line of input will contain a single integer `n` that indicates the dimension of the matrix. Each of the next `n` lines will contain `n` integer numbers separated by one or more spaces.

Output

You will print out the unwrapped spiral of numbers with 10 numbers to a line, where each number is followed by a single space. The last line may have less than 10 numbers.

Example Input File

```
5
11 32 23 34 25
26 17 38 29 10
31 12 13 14 15
36 37 18 19 20
21 22 33 24 25
```

Example Output to Screen

```
11 32 23 34 25 10 15 20 25 24
33 22 21 36 31 26 17 38 29 14
19 18 37 12 13
```

11. Words

Program Name: Words.java   Input File: words.dat

Words is a popular game that people play with others on their smart phones. Since you have been playing this game, you have found many lists of words that you can use to help you win. However, this is not the easiest way to find the word you need. One problem with your list is that the words are not in alphabetical order and they are written across the page with many words on the same line. Another problem is that when you need a word that, for example, has 5 letters and ends in the letter `n`, the list is hard to read. Since you already have a database of words, you are going to write a program that will create an alphabetical list of words that meet the criteria that you need at a given point in the game.

Input

The first line of input will contain a single integer `n` that indicates the number of lines of words in your "dictionary". Each of the next `n` lines will contain an alphabetical list of words with a single space between words. The next line will contain a single integer m that indicates the number of lists that you will generate. Each of the next `m` lines will contain an integer `r`, $2 \leq r \leq 7$, that indicates the length of the word that you need and a space followed by the last letter of the word that you need.

Output

In alphabetical order and one word per line, you will print the words from your dictionary that meet the criteria input. Print `NONE` if there are no words that meet the criteria given. Print a blank line after each list.

Example Input File

```
12
BEEBEE BEEFED BEEPED BEEPER BEETLE BEEVES BEEZER BENDEE BESEEM
BEWEEP BREEZE CHEESE CREESE DECREE DEEDED DEEMED DEEPEN DEEPER
DEGREE DELETE EELIER EERIER EFFETE EKUELE ELEVEN EMCEED EMCEES
EMEERS EMERGE EMESES EMEUTE ENTRÉE EPEE GAGED GAGER GAGES GANGS
GAUGE GIGAS GIGHE GIGOT GIGUE GLOGG GLUGS GOGOS GOING GONGS
GORGE GOUGE GREGO GRIGS GROGS GULAG GURGE HOGGS JAGGS JAGGY
LEGGY LOGGY MIGGS MOGGY MUGGS MUGGY NAGGY NOGGS UREA UVEA ZOEA
RAGGY SAGGY SOGGY VUGGS VUGGY WIGGY YEGGS AGOG EGGS EGGY GAGA
GAGE GAGS GANG GIGA GIGS GLEG GLUG GOGO GONG GRIG GROG HOGG JAGG
MIGG MUGG NOGG VUGG YEGG EGG GAG GIG BEE CEE DEE EEL EKE EME ERE
EVE EWE EYE FEE INBY INCH INFO INIA INKS INKY INLY INNS INRO
INTI INTO INK INN INS EAU AA AE AI OE RAIA ROUE
2
4 Z
5 Y
```

Example Output to Screen

```
NONE

JAGGY
LEGGY
LOGGY
MOGGY
MUGGY
NAGGY
RAGGY
SAGGY
SOGGY
VUGGY
WIGGY
```

12. Word Search

Program Name: WordSearch.java   Input File: wordsearch.dat

Given an *n* by *n* grid of letters, and a list of words, find the location in the grid where the word can be found. A word matches a straight, contiguous line of letters in the grid. The match could either be done horizontally (left or right) or vertically (up or down) or diagonally (lower right to upper left or upper right to lower left or lower left to upper right or upper left to lower right).

Input

The first line of input will contain a single integer `n` that indicates the dimension of the grid of letters. Each of the next `n` lines will contain `n` upper case characters separated by one or more spaces. The next line will contain a single integer `m` that indicates the number of words to find. Each of the next `m` lines will contain a single word in upper case. A word may exist either 0 or 1 times in the grid.

Output

You will print each word on a line and on the same line print the row and the column of the first letter of the word in the grid. The numbering for both the row and the column starts with 1. If the word does not exist in the grid, use 0 for both the row and column number.

Example Input File

```
14
S I N S N O W S T O R M W I
K L N T S N O W M A N E F E
A R E C A L P E R I F R I E
T C G I C A N D L E E R M K
E F H B G T A O C E E Y I A
S R R I L H T U Z P A C T L
T A N O L I B E R I G H T F
B C D H S L Z E T A N R E W
O S D E E T T Z L D R I N O
O E D V L N I S A L S S S N
T B O L I S C Y Y R S T E S
S H H W O L E L O W D M C A
S N A D E C E M B E R A D L
E L T I S E A S O N G S H T
19
FROST
COLD
BOOTS
CANDLE
COAT
SLEIGHBELLS
BLIZZARD
SANTA
FREEZE
HILL
SONGS
DECEMBER
FIREPLACE
MITTENS
SEASON
SKATES
SNOWMAN
TREE
SNOWSTORM
```

12. Word Search (cont.)

Example Output to Screen

```
FROST 5 2
COLD 13 6
BOOTS 8 1
CANDLE 4 5
COAT 5 9
SLEIGHBELLS 1 1
BLIZZARD 5 4
SANTA 0 0
FREEZE 2 13
HILL 5 3
SONGS 14 8
DECEMBER 13 4
FIREPLACE 3 11
MITTENS 4 13
SEASON 14 5
SKATES 1 1
SNOWMAN 2 5
TREE 0 0
SNOWSTORM 1 4
```