7th Annual Manitoba Programming Contest
September 28, 2013

General Instructions:

1. Submit solutions using the PC^2 software.
2. All input should be read from standard input.

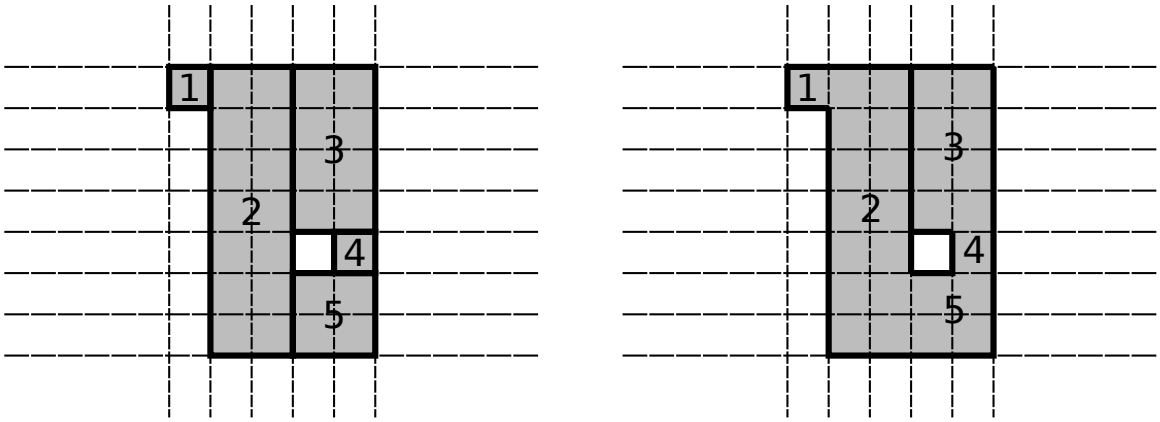
UNIVERSITY OF MANITOBA

Problem 1 – Shipping Container Housing

The latest trend in housing is living in shipping containers! You are part of a company that builds custom houses from shipping containers. Your company has several different sizes of shipping containers, which all have rectangular floors. All shipping containers have the same height. Customers can choose any number of shipping containers, lay them out however they want, and then have the resulting collection built into a house. Your company does insist, however, that all of the containers must be placed in certain ways:

1. the shipping must be laid out on a grid, with their rectangular floors placed within some number of grid cells, in a non-overlapping way.
2. each container in the floor plan must be connected to another container in the floor plan.

Your job is making the doorways in the houses. Any time two containers touch along an edge of the grid, a doorway can be placed there (this means, in particular, that containers that touch only at a corner are not considered to be touching and no doorway can be placed there). A doorway between two containers takes up the entire size of the space the two containers touch, so the cost of placing a door between two containers is the length of the side that they share.



Consider the container configuration on the left, which consists of five containers. Doorways can be placed between, for example, containers 1 and 2, containers 2 and 3, containers 4 and 5, but not between containers 2 and 4, which do not touch. The cost of placing a doorway between containers 2 and 3 would be 4 (the length of the side they share). The configuration on the right shows the result of cutting doors to connect all containers. The cost of these doorways would be 5.

The customers always want the cheapest option for installing doorways in their houses, but there always must be a way to travel from any container the house to any other container. Given a description of the containers in a house, find the minimum cost for installing doorways.

Input

Each input file has multiple house configurations. The input for a house begins with a single integer on its own line. This number, n, is the number of containers in the current house. Each of the next n lines contains the specification for a single container. A container is specified by integers `x1 y1 x2 y2`, which specify `(x1,y1)`, the coordinates of the northwest corner of the container, and `(x2,y2)`, the coordinates of the southeast corner of the container. Houses are guaranteed to satisfy the two rules about layout specified in the question. The position `(0, 0)` is in the northwest corner of the grid. The integers `x1,x2,y1,y2` all satisfy $0 \le x1,x2,y1,y2 \le 10,000$, and n satisfies $1 \le n \le 10,000$.

The last line of the input file is a single integer `0`. This number signifies the end of the input and should not be processed.

Output

For each house, output the minimal cost of constructing the doorways in the house. Use the format "`House #X: c`" where `c` is the cost of the house and `X` is the number of the house, starting with 1 for the first house.

Sample Input	Sample Output
5 0 0 1 1 1 0 3 7 3 0 5 4 4 4 5 5 3 5 5 7 8 0 0 8 1 0 1 1 7 3 1 8 2 3 2 8 3 3 3 8 4 3 4 8 5 3 5 8 7 1 6 3 7 0	House #1: 5 House #2: 23

Problem 2 – Swimming Lane Assignment

You are responsible for scheduling a swim meet between two rival university swim teams, the Dooms and the Monties. After arranging it so that the swimmers from the two teams were in alternating rows, you found out that both teams were unhappy with your lane assignments, and each wanted to be closer to their fans, who sit on one side of the pool or the other.

The lanes of the pool are number 1 through n for some even number n. The Dooms (previously in odd numbered lanes) all want to be in higher number lanes and the Monties (previously in even numbered lanes) want to be in lower number lanes.

You agree to change the lane assignents, but don't guarantee that each swimmer will be as close as possible to their intended side of the pool. You only guarantee that swimmers from the Dooms will get new lanes that will never decrease (they may stay the same) and swimmers from the Monties will get new lanes that will always decrease. You want to know, for a given configuration of the swimmers, how many possible rearrangements of the swimmers you could have subject to the constraints.

For instance, suppose we have four racers, originally assigned to lanes as AbCd. Swimmers A and C are from the Dooms (lanes 1 and 3) and swimmers b and d are from the Monties (lanes 2 and 4). Then the three possible rearrangements are:

bAdC bdAC bdCA

Notice that in each case, the Dooms (A,C) are in a lane number that has increased (or stayed the same). The Monties (b,d) are in a lane number that has decreased.

Input

The input will consist of integers n, each on their own line, where n is even and $0<n\leq16$. The last line of the input is a zero, and should not be processed.

Output

For each input integer n, output n followed by a single space, then the number of possible rearrangements.

Sample Input	Sample Output
4 8 0	4 3 8 155

Problem 3 – Wrapping Ragged Text

One way that documents lay out paragraph text is using ragged right alignment (also known as left-aligned text), where the left edge of the text is aligned in a vertically straight line, but the right edge ends where the words do. Just like the page you're reading now.

When text is laid out in this fashion, it is word-wrapped so that adjacent words are placed across one line at a time until there is no room for the next word, and it is placed at the start of the next line. The lines are filled this way until all the text is laid out.

For example, consider the following text, laid out with in a column size of 19.

```
|Exploring the zoo, |
|we saw every       |
|kangaroo jump and  |
|quite a few carried|
|babies. Expect     |
|skilled signwriters|
|to use many jazzy, |
|quaint old         |
|alphabets          |
|effectively.       |
```

Write a program that takes description of a paragraph of text, a column size, and reports the number of lines required to word-wrap the text.

Input

The first line of input contains an integer n that indicates the number of paragraphs to be wrapped. Each paragraph is described by two lines of input. The first line is a single integer w describing the number of columns, where $1 \le w \le 1000$. The second line is a sequence of integers separated by a single space where each integer is the length of the next consecutive word in the paragraph. Word lengths l will satisfy $1 \le l \le 1000$ and $l \le w$. The number of words on a line will always be between 1 and 10000. Punctuation is included in word length. When typeset, there should be one space between words on the same line.

Output

For each paragraph, output the paragraph number (counting from 1) and the number of lines required to typeset the paragraph, in the form "Paragraph n

requires m lines.". Always output "lines" even if there is only one line.

Sample Input	Sample Output
2 19 9 3 4 2 3 5 8 4 3 5 1 3 7 7 6 7 11 2 3 4 6 6 3 9 12 52 7 8 9 4 6 6 4 5 6	Paragraph 1 requires 10 lines. Paragraph 2 requires 2 lines.

Problem 4 – Cab Headquarters

A taxicab company is searching for a location for their new headquarters. The company has identified a set of potential locations for the headquarters and also has its historical record of how many pick-ups (not drop-offs) occurred at each intersection in the city. Your goal is to identify which location for the new potential headquarters gives the minimum total distance travelled to make all trips from the headquarters location to each of the historical pick-up locations.

Making things easier, the city that the company operates in has blocks that form a perfect grid of size n-by-m. Each location (for pickups and for potential headquarters locations) is given by a pair of integers (i,j), where 0≤i<n and 0≤j<m. The position (0,0) is in the northwest corner of the city. The distance between two intersections is the number of blocks travelled in the shortest path between these two intersections, called the Manhattan distance. The Manhattan distance between position (i,j) and (k,l) is |i-k|+|j-l|, where |.| is the absolute value function.

Input

The input consists of several input test cases. The first line of each input test case begins with a line with four integers, n,m,p,h, where n is the width of the city (in blocks), m is the length of the city (in blocks), p is the number of historical pick-up locations and h is the number of potential headquarters locations. The values satisfy 0<n,m<50 and 0<p,h≤n*m.

The next p lines contain three integers each, i,j and c. This reflects that c pickups have taken place at the intersection labelled (i,j) with 0≤i<n and 0≤j<m. Each value of c is nonzero and at most 10,000.

The final h lines of the test case contains two integers each, f and g, where 0≤f<n and 0≤g<m. This represents that the intersection (f,g) is a potential location for the headquarters.

The final line of the input file, after all test cases, is 0 0 0 0. This line should not be processed as an input test case.

Output

For each test case, output one line

```
Case X: best location at x y with cost C
```

where X is the case number (starting at one with the first test case), (x,y) is the optimal location for the headquarters, and C is the cost of placing the headquarters at that location, i.e., the total amount of distance to travel to all historical pickup locations from that headquarters location.

Sample Input	Sample Output
5 4 4 2 0 0 5 4 2 2 2 1 3 4 0 6 1 1 2 3 4 3 2 2 0 0 10000 3 0 1 1 1 2 1 0 0 0 0	Case 1: best location at 1 1 with cost 45 Case 2: best location at 1 1 with cost 20003

Problem 5 – Lossy Quilting

Charlotte sews quilts from tiles made from her friends and family. She cuts squares of fabric and sends one square to each person. The person decorates the tile in some way, and then returns the square to Charlotte, who connects the tiles into a rectangular shape and produces a quilt.

Unfortunately, Charlotte's friends and family suffer from several deficiencies, including laziness (never finishing their tiles), forgetfulness (misplacing their tiles and not being able to find them) and excessive trust in Canada Post (mailing tiles which are subsequently lost). As a result, Charlotte often starts out with fewer tiles than she started with.

Having produced several quilts in this way, Charlotte has learned who is likely to return tiles and who is not, so she knows the maximum number of tiles that will be lost for any quilt. Charlotte would like to know, given an initial number of tiles (n) and a maximum number of lost tiles (k), whether all possible numbers of tiles that she may receive ($n-k$, $n-k+1$, $n-k+2$,…,n) can be assembled into a rectangle of some length and width, both of which must be at least two. If so, we say that the number n is k-quiltable. Otherwise, we say the number n is not k-quiltable.

For example, 28 is 4-quiltable, since 28,27,26,25 and 24 tiles can all be arranged into rectangles: 28 tiles make a 7x4 rectangle, 27 tiles can be made into a 9x3 rectangle, 26 tiles can be made into a 13x2 rectangle, 25 tiles make a 5x5 rectangle and 24 tiles make a 6x4 rectangle.

Write a program that, given n and k, decides if n is k-quiltable or not.

Input

The first line of the input contains c, the number of test cases to be examined. The next c lines each contain two integers, n and k. The values of n and k satisfy $2 \le n \le 2^{32}$ and $1 \le k \le 100$.

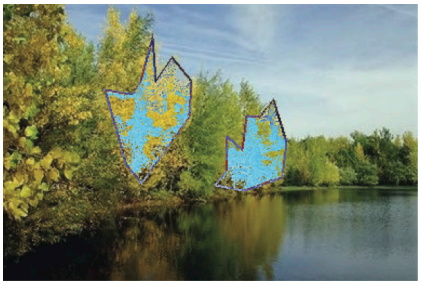
Output

For each input case, write either "`n is not k-quiltable.`" or "`n is k-quiltable.`" based on the criteria above.

Sample Input	Sample Output
4 28 4 10 3 10 2 89752 62	28 is 4-quiltable. 10 is not 3-quiltable. 10 is 2-quiltable. 89752 is 62-quiltable.

Problem 6 – Pixel Similarity

You have been asked to devise a way to measure the similarity between regions of a digital image. One way to do this is to find all the pixels in the two regions that have similar colours (as shown in the image below).



The term *similar* can be quantified by considering the Euclidean distance between pixel RGB values and choosing a threshold to determine the degree of similarity. Recall, the Euclidean distance between pixels $p_1 = (R_1,G_1,B_1)$ and $p_2 = (R_2,G_2,B_2)$ is defined as $\sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$.

Your job, given a parameter ε and two sets A and B, is to find all the pixels in A and B that that have a colour whose Euclidean distance to a pixel colour in the opposite set is less than or equal to ε.

Input

The first line of input contains the number of test cases in the file. All test cases then follow.

Each test case starts with a line with five integers in it: ε, s_A, c_A, s_B, c_B. The parameter ε is the maximum distance between any two pixels that should be considered similar. The integers s_A and s_B are the number of pixels in the sets A and B, while c_A and c_B are the maximum number of different pixel values in the sets A and B. The value ε satisfies $0 \le \varepsilon \le 256$, while $0 \le s_A, s_B < 50{,}000$ and $0 \le c_A, c_B < 1{,}000$.

The remaining $s_A + s_B$ lines of the test case follow, each with three integer values R G B on one line, representing the RGB value for one pixel. Set A is made up of the first s_A lines and the set B is the last s_B lines. The values R,G and B are each in the range 0-255.

Output

For each test case, first write "Case #x" on a line, starting with case number 1. Following this, write " Set A" on a line (note one leading space) and then the indices of each of the pixels in A that are similar to some pixel in B. Give the index of each pixel on a separate line, preceeded by two spaces. Pixels should be indexed starting from 0 as they appear in the input. After this, give a similar description of B: the line " Set B", followed by one line for each index of a pixel in B that is similar to some pixel in A. Leave a blank line after each test case.

Sample Input	Sample Output

1
2 11 10 7 7
1 3 0
1 3 0
2 2 0
1 1 0
0 9 0
0 0 0
0 0 3
2 0 1
2 2 1
9 6 5
4 2 3
0 1 0
1 3 0
0 1 0
6 8 7
5 7 6
2 4 1
0 3 0
	Case #1
	 Set A
	  0
	  1
	  2
	  3
	  5
	  8
	 Set B
	  0
	  1
	  2
	  5
	  6