ACM International Collegiate Programming Contest
1999 East Central Regional Contest
University of Waterloo
November 13, 1999

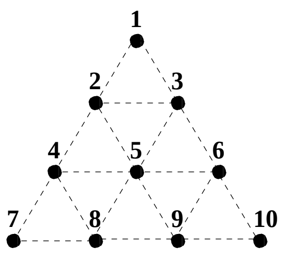
Sponsored by IBM

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output.
    - No whitespace should appear at the end of a line, and all lines should be terminated with a new-line.
    - Tabs should never be used.
    - Output must correspond *exactly* to the provided sample output, including (mis)spelling and spacing. Multiple spaces will not be used in any of the judges' output, except where expressly stated.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the submit command.
6. Each solution (C, C++, Pascal) is given 5 CPU seconds and 16 MB of memory to run for each input file. The run time and memory limits on Java solutions will be given separately in another document.
7. The allowed programming languages are C, C++, Java, and Pascal.
8. Judges' decisions are to be considered final. No cheating will be tolerated.
9. There are **eight** questions to be completed in **five hours**.

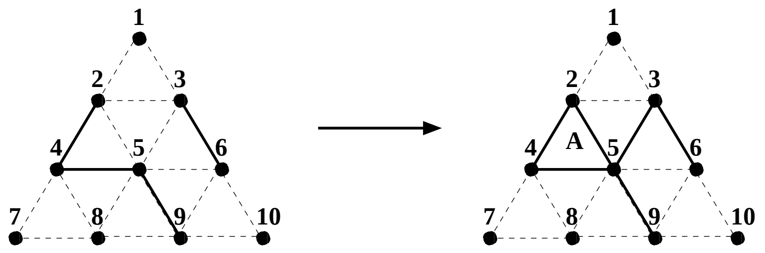
Problem A: Triangle War

Triangle War is a two-player game played on the following triangular grid:



Two players, A and B, take turns filling in any dotted line connecting two dots, with A starting first. Once a line is filled, it cannot be filled again. If the line filled by a player completes one or more triangles, she owns the completed triangles and she is awarded another turn (i.e. the opponent skips a turn). The game ends after all dotted lines are filled in, and the player with the most triangles wins the game. The difference in the number of triangles owned by the two players is not important.

For example, if A fills in the line between 2 and 5 in the partial game on the left below:



Then, she owns the triangle labelled A and takes another turn to fill in the line between 3 and 5. B can now own 3 triangles (if he wishes) by filling in the line between 2 and 3, then the one between 5 and 6, and finally the one between 6 and 9. B would then make one more move before it is A's turn again.

In this problem, you are given a number of moves that have already been made. From the partial game, you should determine which player will win assuming that each player plays a perfect game from that point on. That is, assume that each player always chooses the play that leads to the best possible outcome for himself/herself.

Input

You will be given a number of games in the input. The first line of input is a positive integer indicating the number of games to follow. Each game starts with an integer $6 \leq m \leq 18$ indicating the number of moves that have been made in the game. The next m lines indicate the moves made by the two players in order, each of the form $i \quad j$ (with $i < j$) indicating that the line between i and j is filled in that move. You may assume that all given moves are legal.

Output

For each game, print the game number and the result on one line as shown below. If A wins, print the sentence "A wins." If B wins, print "B wins."

Sample Input

```
4
6
2 4
4 5
5 9
3 6
2 5
3 5
7
2 4
4 5
5 9
3 6
2 5
3 5
7 8
6
1 2
2 3
1 3
2 4
2 5
4 5
10
1 2
2 5
3 6
5 8
4 7
6 10
2 4
4 5
4 8
7 8
```

Sample Output

```
Game 1: B wins.
Game 2: A wins.
Game 3: A wins.
Game 4: B wins.
```

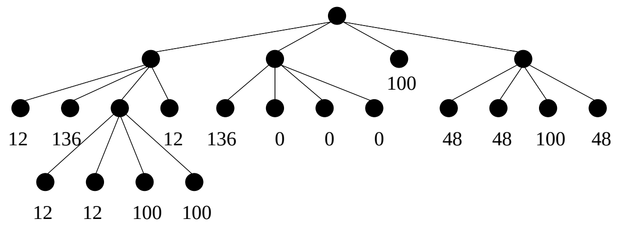
Problem B: Unscrambling Images

Quadtrees are commonly used for encoding digital images in a compact form. Given an $n \times n$ image (where n is a power of 2, $1 \leq n \leq 16$), its quadtree encoding is computed as follows. Start with a quadtree with exactly one node, namely the root, and associate with this node the $n \times n$ square region for the entire image. Then the following is performed recursively:

1. If every pixel in the region associated with the current node has an intensity value of p, then the node is made a leaf and it is assigned an associated value of p.
2. Otherwise, four nodes are added as children of the current node. The region is divided into four equal (square) quadrants and each quadrant is assigned to one child node. The algorithm recurses on each of the children nodes.

When the process terminates, we obtain a quadtree in which every internal node has four children. Every leaf node has an associated value representing the intensity of the region corresponding to the leaf node. An example of an image and its quadtree encoding is shown below.

12	12	136	136	136	136	0	0
12	12	136	136	136	136	0	0
12	12	12	12	0	0	0	0
100	100	12	12	0	0	0	0
100	100	100	100	48	48	48	48
100	100	100	100	48	48	48	48
100	100	100	100	100	100	48	48
100	100	100	100	100	100	48	48



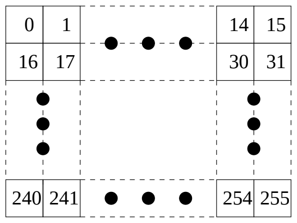
We have assumed that the four children represent, from left to right, the upper left, upper right, lower left, and lower right quadrants, respectively.

To easily identify a node in a quadtree, we assign a number to each node by the following rules:

1. The root is numbered 0.
2. If the number of a node is k, then its children, from left to right, are numbered $4k+1, 4k+2, 4k+3, 4k+4$.

Images encoded as quadtrees can be encrypted by a secret password as follows: whenever a subdivision is performed, the four branches are reordered. The reordering may be different at each node, but it is completely determined by the password and the node number.

Unfortunately, some people use the "save password" feature in the encoding program and use the same password for multiple images. By observing the encoding of a well-chosen test image, any image encoded with the same password can be decoded without the password. In this test image, each pixel has a distinct intensity from 0 to $n^2 - 1$ arranged from left-to-right, top-to-bottom in increasing order. An example for $n = 16$ is given below:



You managed to gain access to the encoding program and used it to encode the test image. Given the quadtree encoding of the test image, write a program to decode any other image encoded with the same password.

Input

You will be given a number of cases in the input. The first line of input consists of a positive integer indicating the number of test cases to follow. Each test case starts with a line containing n, followed by the quadtree encoding of the test image and the quadtree encoding of the secret image to be decoded. Each quadtree encoding starts with a line containing a positive integer m indicating the number of leaf nodes in the tree. The next m lines are of the form:

```
k intensity
```

which specifies that the node numbered k is a leaf node with the specified intensity as the associated leaf value. Nodes not specified are either internal nodes or absent in the quadtree. You may assume that all intensities are between 0 and 255, inclusively. You may also assume that each quadtree encoding is a valid output obtained from the encoding algorithm described above.

Output

For each test case, print the case number followed by a blank line. Then, print the intensities of the pixels of the decoded image one row at a time. Print each intensity right-justified in a field of width 4, with no extra spaces between fields. Insert a blank line between cases.

Sample Input

```
2
2
4
1 3
2 2
3 0
4 1
4
1 23
2 123
3 253
4 40
4
```

```
16
5 8
6 9
7 13
8 12
9 0
10 4
11 1
12 5
13 2
14 3
15 7
16 6
17 10
18 11
19 15
20 14
7
2 10
3 20
4 30
5 41
6 42
7 44
8 43
```

Sample Output

```
Case 1

 253  40
 123  23

Case 2

  10  10  20  20
  10  10  20  20
  41  42  30  30
  43  44  30  30
```

Problem C: A Plug for UNIX

You are in charge of setting up the press room for the inaugural meeting of the United Nations Internet eXecutive (UNIX), which has an international mandate to make the free flow of information and ideas on the Internet as cumbersome and bureaucratic as possible.

Since the room was designed to accommodate reporters and journalists from around the world, it is equipped with electrical receptacles to suit the different shapes of plugs and voltages used by appliances in all of the countries that existed when the room was built. Unfortunately, the room was built many years ago when reporters used very few electric and electronic devices and is equipped with only one receptacle of each type. These days, like everyone else, reporters require many such devices to do their jobs: laptops, cell phones, tape recorders, pagers, coffee pots, microwave ovens, blow dryers, curling irons, tooth brushes, etc. Naturally, many of these devices can operate on batteries, but since the meeting is likely to be long and tedious, you want to be able to plug in as many as you can.

Before the meeting begins, you gather up all the devices that the reporters would like to use, and attempt to set them up. You notice that some of the devices use plugs for which there is no receptacle. You wonder if these devices are from countries that didn't exist when the room was built. For some receptacles, there are several devices that use the corresponding plug. For other receptacles, there are no devices that use the corresponding plug.

In order to try to solve the problem you visit a nearby parts supply store. The store sells adapters that allow one type of plug to be used in a different type of outlet. Moreover, adapters are allowed to be plugged into other adapters. The store does not have adapters for all possible combinations of plugs and receptacles, but there is essentially an unlimited supply of the ones they do have.

Input

The input will consist of one case. The first line contains a single positive integer n ($1 \leq n \leq 100$) indicating the number of receptacles in the room. The next n lines list the receptacle types found in the room. Each receptacle type consists of a string of at most 24 alphanumeric characters. The next line contains a single positive integer m ($1 \leq m \leq 100$) indicating the number of devices you would like to plug in. Each of the next m lines lists the name of a device followed by the type of plug it uses (which is identical to the type of receptacle it requires). A device name is a string of at most 24 alphanumeric characters. No two devices will have exactly the same name. The plug type is separated from the device name by a space. The next line contains a single positive integer k ($1 \leq k \leq 100$) indicating the number of different varieties of adapters that are available. Each of the next k lines describes a variety of adapter, giving the type of receptacle provided by the adapter, followed by a space, followed by the type of plug.

Output

A line containing a single non-negative integer indicating the smallest number of devices that cannot be plugged in.

Sample Input

```
4
A
B
C
D
5
laptop B
phone C
pager B
clock B
comb X
3
B X
X A
X D
```

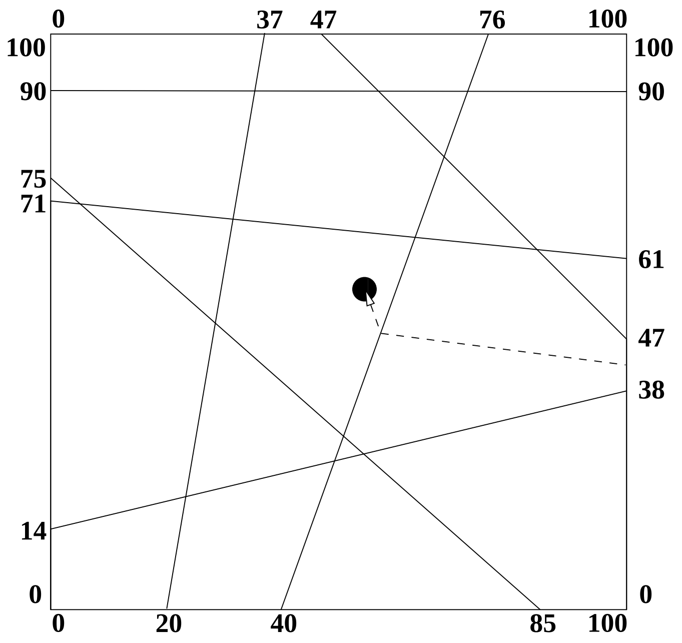
Sample Output

```
1
```

Problem D: Treasure Hunt

Archeologists from the Antiquities and Curios Museum (ACM) have flown to Egypt to examine the great pyramid of Key-Ops. Using state-of-the-art technology they are able to determine that the lower floor of the pyramid is constructed from a series of straightline walls, which intersect to form numerous enclosed chambers. Currently, no doors exist to allow access to any chamber. This state-of-the-art technology has also pinpointed the location of the treasure room. What these dedicated (and greedy) archeologists want to do is blast doors through the walls to get to the treasure room. However, to minimize the damage to the artwork in the intervening chambers (and stay under their government grant for dynamite) they want to blast through the minimum number of doors. For structural integrity purposes, doors should only be blasted at the midpoint of the wall of the room being entered. You are to write a program which determines this minimum number of doors.

An example is shown below:



Input

The input will consist of one case. The first line will be an integer n ($0 \leq n \leq 30$) specifying number of interior walls, followed by n lines containing integer endpoints of each wall x_1 y_1 x_2 y_2. The 4 enclosing walls of the pyramid have fixed endpoints at $(0,0), (0,100), (100,100)$ and $(100,0)$ and are *not* included in the list of walls. The interior walls always span from one exterior wall to another exterior wall and are arranged such that no more than two walls intersect at any point. You may assume that no two given walls coincide. After the listing of the interior walls there will be one final line containing the floating point coordinates of the treasure in the treasure room (guaranteed not to lie on a wall).

Output

Print a single line listing the minimum number of doors which need to be created, in the format shown below.

Sample Input

```
7
20 0 37 100
40 0 76 100
85 0 0 75
100 90 0 90
0 71 100 61
0 14 100 38
100 47 47 100
54.5 55.4
```

Sample Output

```
Number of doors = 2
```

Problem E: 487-3279

Businesses like to have memorable telephone numbers. One way to make a telephone number memorable is to have it spell a memorable word or phrase. For example, you can call the University of Waterloo by dialing the memorable TUT–GLOP. Sometimes only part of the number is used to spell a word. When you get back to your hotel tonight you can order a pizza from Gino's by dialing 310–GINO. Another way to make a telephone number memorable is to group the digits in a memorable way. You could order your pizza from Pizza Hut by calling their "three tens" number 3–10–10–10.

The standard form of a telephone number is seven decimal digits with a hyphen between the third and fourth digits (e.g. 888–1200). The keypad of a phone supplies the mapping of letters to numbers, as follows:

A, B, and C map to 2
D, E, and F map to 3
G, H, and I map to 4
J, K, and L map to 5
M, N, and O map to 6
P, R, and S map to 7
T, U, and V map to 8
W, X, and Y map to 9

There is no mapping for Q or Z. Hyphens are not dialed, and can be added and removed as necessary. The standard form of TUT–GLOP is 888–4567, the standard form of 310–GINO is 310–4466, and the standard form of 3–10–10–10 is 310–1010.

Two telephone numbers are equivalent if they have the same standard form. (They dial the same number.)

Your company is compiling a directory of telephone numbers from local businesses. As part of the quality control process you want to check that no two (or more) businesses in the directory have the same telephone number.

Input

The input will consist of one case. The first line of the input specifies the number of telephone numbers in the directory (up to 100,000) as a positive integer alone on the line. The remaining lines list the telephone numbers in the directory, with each number alone on a line. Each telephone number consists of a string composed of decimal digits, uppercase letters (excluding Q and Z) and hyphens. Exactly seven of the characters in the string will be digits or letters.

Output

Generate a line of output for each telephone number that appears more than once in any form. The line should give the telephone number in standard form, followed by a space, followed by the number of times the telephone number appears in the directory. Arrange the output lines by telephone number in ascending lexicographical order. If there are no duplicates in the input print the line:

```
No duplicates.
```

Sample Input

```
12
4873279
ITS-EASY
888-4567
3-10-10-10
888-GLOP
TUT-GLOP
967-11-11
310-GINO
F101010
888-1200
-4-8-7-3-2-7-9-
487-3279
```

Sample Output

```
310-1010 2
487-3279 4
888-4567 3
```

Problem F: Biorhythms

Some people believe that there are three cycles in a person's life that start the day he or she is born. These three cycles are the physical, emotional, and intellectual cycles, and they have periods of lengths 23, 28, and 33 days, respectively. There is one peak in each period of a cycle. At the peak of a cycle, a person performs at his or her best in the corresponding field (physical, emotional or mental). For example, if it is the mental curve, thought processes will be sharper and concentration will be easier.

Since the three cycles have different periods, the peaks of the three cycles generally occur at different times. We would like to determine when a triple peak occurs (the peaks of all three cycles occur in the same day) for any person. For each cycle, you will be given the number of days from the beginning of the current year at which one of its peaks (not necessarily the first) occurs. You will also be given a date expressed as the number of days from the beginning of the current year. You task is to determine the number of days from the given date to the next triple peak. The given date is not counted. For example, if the given date is 10 and the next triple peak occurs on day 12, the answer is 2, not 3. If a triple peak occurs on the given date, you should give the number of days to the next occurrence of a triple peak.

Input

You will be given a number of cases. The input for each case consists of one line of four integers p, e, i, and d. The values p, e, and i are the number of days from the beginning of the current year at which the physical, emotional, and intellectual cycles peak, respectively. The value d is the given date and may be smaller than any of p, e, or i. All values are non-negative and at most 365, and you may assume that a triple peak will occur within 21252 days of the given date. The end of input is indicated by a line in which $p = e = i = d = -1$.

Output

For each test case, print the case number followed by a message indicating the number of days to the next triple peak, in the form:

```
Case 1: the next triple peak occurs in 1234 days.
```

Use the plural form "days" even if the answer is 1.

Sample Input

```
0 0 0 0
0 0 0 100
5 20 34 325
4 5 6 7
283 102 23 320
203 301 203 40
-1 -1 -1 -1
```

Sample Output

```
Case 1: the next triple peak occurs in 21252 days.
Case 2: the next triple peak occurs in 21152 days.
Case 3: the next triple peak occurs in 19575 days.
Case 4: the next triple peak occurs in 16994 days.
Case 5: the next triple peak occurs in 8910 days.
Case 6: the next triple peak occurs in 10789 days.
```

Problem G: Gone Fishing

John is going on a fishing trip. He has h hours available ($1 \leq h \leq 16$), and there are n lakes in the area ($2 \leq n \leq 25$) all reachable along a single, one-way road. John starts at lake 1, but he can finish at any lake he wants. He can only travel from one lake to the next one, but he does not have to stop at any lake unless he wishes to. For each $i = 1, \ldots, n-1$, the number of 5-minute intervals it takes to travel from lake i to lake $i+1$ is denoted t_i ($0 < t_i \leq 192$). For example, $t_3 = 4$ means that it takes 20 minutes to travel from lake 3 to lake 4.

To help plan his fishing trip, John has gathered some information about the lakes. For each lake i, the number of fish expected to be caught in the initial 5 minutes, denoted f_i ($f_i \geq 0$), is known. Each 5 minutes of fishing decreases the number of fish expected to be caught in the next 5-minute interval by a constant rate of d_i ($d_i \geq 0$). If the number of fish expected to be caught in an interval is less than or equal to d_i, there will be no more fish left in the lake in the next interval. To simplify the planning, John assumes that no one else will be fishing at the lakes to affect the number of fish he expects to catch.

Write a program to help John plan his fishing trip to maximize the number of fish expected to be caught. The number of minutes spent at each lake must be a multiple of 5.

Input

You will be given a number of cases in the input. Each case starts with a line containing n. This is followed by a line containing h. Next, there is a line of n integers specifying f_i ($1 \leq i \leq n$), then a line of n integers d_i ($1 \leq i \leq n$), and finally, a line of $n-1$ integers t_i ($1 \leq i \leq n-1$). Input is terminated by a case in which $n = 0$.

Output

For each test case, print the number of minutes spent at each lake, separated by commas, for the plan achieving the maximum number of fish expected to be caught (you should print the entire plan on one line even if it exceeds 80 characters). This is followed by a line containing the number of fish expected. If multiple plans exist, choose the one that spends as long as possible at lake 1, even if no fish are expected to be caught in some intervals. If there is still a tie, choose the one that spends as long as possible at lake 2, and so on. Insert a blank line between cases.

Sample Input

```
2
1
10 1
2 5
2
4
4
10 15 20 17
0 3 4 3
1 2 3
4
4
10 15 50 30
0 3 4 3
1 2 3
0
```

Sample Output

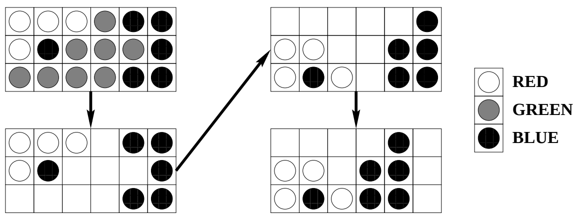
```
45, 5
Number of fish expected: 31

240, 0, 0, 0
Number of fish expected: 480

115, 10, 50, 35
Number of fish expected: 724
```

Problem H: The Same Game

The game named "Same" is a single-person game played on a 10×15 board. Each square contains a ball colored red (R), green (G), or blue (B). Two balls belong to the same cluster if they have the same color, and one can be reached from another by following balls of the same color in the four directions up, down, left, and right. At each step of the game, the player chooses a ball whose cluster has at least two balls and removes all balls in the cluster from the board. Then, the board is "compressed" in two steps:

1. Shift the remaining balls in each column down to fill the empty spaces. The order of the balls in each column is preserved.
2. If a column becomes empty, shift the remaining columns to the left as far as possible. The order of the columns is preserved.

For example, choosing the ball at the bottom left corner in the sub-board below causes:



The objective of the game is to remove every ball from the board, and the game is over when every ball is removed or when every cluster has only one ball. The scoring of each game is as follows. The player starts with a score of 0. When a cluster of m balls is removed, the player's score increases by $(m-2)^2$. A bonus of 1000 is given if every ball is removed at the end of the game.

You suspect that a good strategy might be to choose the ball that gives the largest possible cluster at each step, and you want to test this strategy by writing a program to simulate games played using this strategy. If there are two or more balls to choose from, the program should choose the leftmost ball giving the largest cluster. If there is still a tie, it should choose the bottommost ball of these leftmost balls.

Input

You will be given a number of games in the input. The first line of input contains a positive integer giving the number of games to follow. The initial arrangement of the balls of each game is given one row at a time, from top to bottom. Each row contains 15 characters, each of which is one of "R", "G", or "B", specifying the colors of the balls in the row from left to right. A blank line precedes each game.

Output

For each game, print the game number, followed by a new line, followed by information about each move, followed by the final score. Each move should be printed in the format:

```
Move x at (r,c): removed b balls of color C, got s points.
```

where x is the move number, r and c are the row number and column number of the chosen ball, respectively. The rows are numbered from 1 to 10 from the bottom, and columns are numbered from 1 to 15 from the left. b is the number of balls in the cluster removed. C is one of "R", "G", or "B", indicating the color of the balls removed. s is the score for this move. The score does not include the 1000 point bonus if all the balls are removed after the move.

The final score should be reported as follows:

```
Final score: s, with b balls remaining.
```

Insert a blank line between the output of each game. Use the plural forms "balls" and "points" even if the corresponding value is 1.

Sample Input

```
3

RGGBBGGRBRRGGBG
RBGRBGRBGRBGRBG
RRRRGBBBRGGRBBB
GGRGBGGBRRGGGBG
GBGGRRRRRBGGRRR
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRGGGGRRRRR
GGGGGGGGGGGGGGG

RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG
BBBBBBBBBBBBBBB
BBBBBBBBBBBBBBB
RRRRRRRRRRRRRRR
RRRRRRRRRRRRRRR
GGGGGGGGGGGGGGG
GGGGGGGGGGGGGGG

RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
BGRBGRBGRBGRBGR
GRBGRBGRBGRBGRB
RBGRBGRBGRBGRBG
```

Sample Output

```
Game 1:

Move 1 at (4,1): removed 32 balls of color B, got 900 points.
Move 2 at (2,1): removed 39 balls of color R, got 1369 points.
Move 3 at (1,1): removed 37 balls of color G, got 1225 points.
Move 4 at (3,4): removed 11 balls of color B, got 81 points.
Move 5 at (1,1): removed 8 balls of color R, got 36 points.
Move 6 at (2,1): removed 6 balls of color G, got 16 points.
Move 7 at (1,6): removed 6 balls of color B, got 16 points.
Move 8 at (1,2): removed 5 balls of color R, got 9 points.
Move 9 at (1,2): removed 5 balls of color G, got 9 points.
Final score: 3661, with 1 balls remaining.

Game 2:

Move 1 at (1,1): removed 30 balls of color G, got 784 points.
Move 2 at (1,1): removed 30 balls of color R, got 784 points.
Move 3 at (1,1): removed 30 balls of color B, got 784 points.
Move 4 at (1,1): removed 30 balls of color G, got 784 points.
Move 5 at (1,1): removed 30 balls of color R, got 784 points.
Final score: 4920, with 0 balls remaining.

Game 3:

Final score: 0, with 150 balls remaining.
```