Polynomial Drawing

If you like polynomials, you might have solved problems like POLEVAL, POLYNOM or POLYTABL. But if you are especially interested in graphs of polynomials, here is the problem for you:
Given a polynomial, your task is to create an image file of its graph.

Input

Input starts with a positive integer t<100, the number of testcases, then t testcases follow. Every testcase contains the representation of a polynomial of at most fourth degree in "pretty printing" notation. Coefficients are either given as integers or fractions. The absolute values of all numbers including numerator and denominator are not larger than 100.

If all coefficients are integers and degree is not larger than 1, a polynomial (which means a testcase) is given in a single line (the *base line*).
If all coefficients are integers, but degree is larger than 1, a polynomial is given in two lines. The first of these contains only the exponents, the second line is the *base line*.
If at least one coefficient is given as a fraction, a polynomial is given in three lines. The first line contains the exponents (if there are any) and the numerators, the last line the denominators.

Polynomials are given in **canonical form**, which means:
⇢ powers of x are given in decreasing order and do not appear more than once
⇢ there is always exactly one operator (either + or -) between consecutive summands
⇢ coefficients with an absolute value of 1 are left out
⇢ summands with coefficients with a value of 0 are left out except for the null polynomial
⇢ there are no whitespaces in the base line except those below exponents

The fraction bar is formed by as many dashes as there are digits in numerator respectively denominator. If the number of digits in numerator and denominator is different, the fraction bar is as long as the larger value; the smaller one is aligned to the right.

```
                        3 4  2        3  10     2
Valid polynomials are:  --x +x       -x+2     5x +---     x -5
                        5            100

                       -3 4  2     2      1             3
Invalid polynomials are:  -x +x       x+3x     x---    4x -13     2x +-5
                        5                7
```

The base line of the polynomial is followed by a single space, then the information about the drawing range is given. It is denoted as "[a;b]", where a and b are integer values with -10 ≤ a < b ≤ 10.

Note: There are no trailing whitespaces and every testcase is followed by a blank line.

Output

For every testcase you have to generate a compressed version of a pgm-imagefile (type P2) showing the polynomial's graph. The first line of output is the pgm-header printed in a single line:

"P2 [w] [h] 2", where [w] and [h] have to be replaced by the width and height of the image (see example below).
The following one to three lines contain the polynomial's term and drawing interval exactly as given in the input. As this would make the pgm-file invalid, each of these lines has to start with a leading "#" followed by a whitespace.

After that h lines of data follow. Every line consists of w values separated by single spaces. Each value is either 0 (black), 1 (gray) or 2 (white) and represents a single pixel of the image.

The width of the image is derived from the given interval [a;b]. For all images we take 20px as size of one unit in x- and y-direction. For every pixel in horizontal direction calculate the correponding y-value and mark that point of the graph by a black pixel. If a calculated y-value does not exactly match to the position of a pixel, you always choose the pixel with least vertical distance. If there is more than one of these pixels, take the pixel that represents the lower value. No (intermediate) value will have an absolute value larger than 10^{18}, if calculations are done carefully.

The height of the image depends on the minimum and maximum values of the polynomial in the given drawing interval, where "value" refers to the y-value of the drawn point, not to the exact y-value of the corresponding x-value if that differs. The range for the y-values reaches from the largest integer smaller than the minimum value to the smallest integer value larger than the maximum value. So, if y_{min}=-3.2 and y_{max}=4 the range for the y-values reaches from -4 to 5 and results in h=181. The value of h will always be less than 1000.

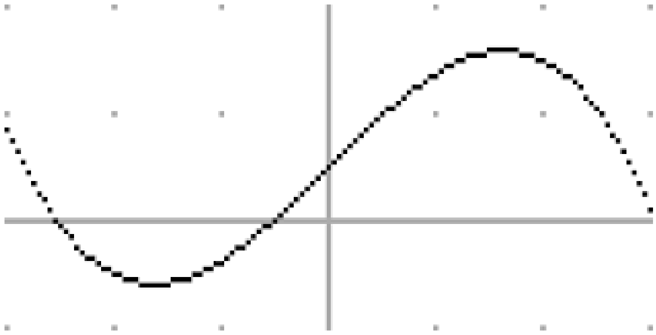
The background of the image is white. All lattice points of the coordinate system are marked with a single gray pixel. If the given x-intervall or the calculated y-range include zero, the x- and/or y-axis are drawn as gray line with 1px linewidth. After drawing the coordinate system including the grid, the graph of the polynomial is drawn. For every pixel in x-direction the corresponding y-value is calculated and the resulting point drawn as a single black pixel.

As this valid pgm image representation produces rather large image files, it has to be printed in **egg-format** (*encoded grayscale graphics*), which uses a two-pass run length encoding compression.

The header gets an additional information about the egg-format: The pgm header line is followed by a single space and an indication of the format "(egg)". The second line of the egg-file is a new additional comment containing information about compression rate: "# Compression: xx.xx%". The compression rate is calculated as "1-(egg-size/pgm-size)", where "size" refers to the pure image size without header, comments and newline characters. The compression rate has to be denoted as percentage with an accurary of two digits after the decimal point.

In *pass one*, every row is encoded. First all spaces between the pixel values are removed. Then every consecutive sequence of the same color that is longer than three pixels is replaced by brackets including the total value of consecutive pixels of that color. For identification of color, different typs of brackets are used: round brackets for white pixels, braces for gray pixels and square brackets for black pixels.
Example: 0 0 0 0 0 2 2 2 2 1 1 1 2 2 1 1 1 1 1 1 1 is encoded to [5](4)11122{7}

In *pass two* consecutive rows are encoded, if there are (at least two) consecutive identical rows. In that case, only the first row is used, followed by the total number of identical consecutive rows in angle brackets "< >".



Unfortunately the egg-format is not widely spread, in fact there is actually no image viewer that is able to handle images in egg-format, except the online egg-viewer. So, if you want to make your egg-files visible, I recommend the egg-viewer: just paste your egg data and make the image visible. The figure on the right shows the graph of the example output (scaled by a factor of 3 for better recognition of single pixels).

Note: Print a blank line after every testcase except the last one.

Example

Input:

```
1
 1 3   1
--x +x+- [-3;3]
 8     2
```

Output:

```
P2 121 61 2 (egg)
# Compression: 93.95%
#  1 3   1
# --x +x+- [-3;3]
#  8     2
1(19)1(19)1(19)1(19)1(19)1(19)1
(60)1(60)<7>
(60)1(29)[6](25)
(60)1(25)[4](6)000(22)
(60)1(23)00(13)00(20)
(60)1(21)00(17)00(18)
(60)1(20)0(21)0(17)
(60)1(18)00(23)00(15)
(60)1(17)0(27)0(14)
(60)1(15)00(29)0(13)
(60)1(14)0(45)
(60)1(13)0(33)0(12)
(60)1(12)0(35)0(11)
(60)1(10)00(37)0(10)
1(19)1(19)1(19)1(9)0(9)1(19)1(10)0(8)1
(60)1(8)0(51)
(60)1(7)0(43)0(8)
0(59)1(6)0(53)
(60)1(5)0(46)0(7)
20(58)1(4)0(48)0(6)
(60)12220(56)
220(57)1220(51)0(5)
(60)120(58)
2220(56)10(54)0(4)
(60)0(60)
(4)0(54)01(56)0222
(58)021(60)
(5)0(51)0221(57)022
```

```
(56)02221(60)
(6)0(48)0(4)1(58)02
(7)0(46)0(5)1(60)
(53)0(6)1(60)
(8)0(43)0(7)1(59)0
(51)0(8)1(60)
{9}0{40}0{70}
(10)0(37)00(10)1(60)
(11)0(35)0(12)1(60)
(12)0(33)0(13)1(60)
(45)0(14)1(60)
(13)0(29)00(15)1(60)
(14)0(27)0(17)1(60)
(15)00(24)0(18)1(60)
(17)0(21)00(19)1(60)
(18)00(17)00(21)1(60)
(20)00(13)00(23)1(60)
(22)000(6)[4](25)1(60)
(25)[6](29)1(60)
(60)1(60)<7>
1(19)1(19)1(19)1(19)1(19)1(19)1
```