# 2142 PeopleFirst

The human rights organization, PeopleFirst, has observers around the world checking up on human rights abuses in various countries. Since some countries take a dim view of this sort of activity, PeopleFirst needs a way for their observers to report abuses without the countries they are residing in realizing that a report is being made. The idea is this: Their observers will create websites where they will display photographs of their sightseeing trips, collections, or other hobbies, but the pictures will be subtly altered to hide the reports in them.

There is software available to do this, but PeopleFirst doesn't want to run the risk that some government will try one of these programs on their observer's pictures and find a hidden message. To avoid that, one of their members has written a custom program for hiding messages in pictures. Unfortunately, she had to leave on her own stint as an observer before she could write the program that extracts the messages from the pictures. That's where you come in. You've been asked to write the missing program.

## Input

The observer's photographs are posted to their web sites in GIF format. Prior to being given to your program, the pictures are run through a filter that converts them to ppm format, so your program has a simpler picture representation to work with. The ppm format begins with the two characters '`P6`' followed by whitespace, most often a single linefeed character. This is followed by the picture width and height formatted as two decimal numbers in ASCII, separated by whitespace. The height is followed by more whitespace, again most often a single linefeed. Following this is the maximum color component value: a decimal number in ASCII. This is the value of the brightest shade for any of the three primary colors in a pixel. You should ignore this value and treat it as if it were 255. This is immediately followed by a single whitespace character, usually a linefeed. Immediately after the whitespace character are $3 \times \mathit{width} \times \mathit{height}$ bytes representing a left to right, top to bottom scan of the pixels in the picture, three bytes per pixel, representing the shades of red, green and blue for the pixel, in that order. Treat each byte as an unsigned binary number representing the shade.

The message is hidden in the picture by altering the original pixel values slightly. Each pixel in the picture encodes one bit of the message, which is extracted by taking the exclusive-or of the low order bits of the shades of red, green and blue that make up the pixel. Message bits are assigned to pixels from left to right in rows, starting with the top left pixel in the picture, and ending with the bottom right pixel. The first thing encoded in the picture is the length of the message. It is encoded as 32 bits of binary, starting with the low order bit. The length is immediately followed by the bytes of the message, each byte encoded with the low order bit first. The message is immediately followed by a 32-bit checksum, with the low order bit appearing first.

The checksum is calculated from all of the bytes of the message length and the message. In calculating the checksum, the bytes of the message length are ordered with the low order byte first. Initialize the checksum to hexadecimal `ffffffff`. For each byte that goes into the checksum, first do a single bit left circular shift of the 32 bits of the checksum, then add the byte as an unsigned value, ignoring any overflow.

## Output

Output consists of the extracted message, or an error message if no hidden message was found in the input picture. A bad message length or invalid checksum indicate that the input picture has no hidden

message.

If the extracted message length is zero, output the following error message and exit before producing any other output:

`No message found: message length is zero`

If the extracted message length has a value so large that there wouldn't be enough pixels in the input picture to encode all of the bits of the message length, message and checksum, output the following error message and exit before producing any other output:

`No message found: bad length:` *msz* `bits >` *psz* `pixels`

In the actual message, replace *msz* with the number of bits needed to encode the message length, message and checksum; replace *psz* with the number of pixels in the picture. Output both numbers in decimal.

If the checksum you calculate does not equal the extracted checksum, output the following error message and exit before producing any other output:

`No message found: bad checksum: recorded =` *rcs*`, calculated =` *ccs*

In the actual message, replace *rcs* with the extracted checksum, and replace *ccs* with the calculated checksum. Output both checksums with the prefix '`0x`', followed by 8 hexadecimal digits, using lower case for the alphabetic hexadecimal digits. The string "`0x13579bdf`" would be one example of the required format.

The judge's data will have no improperly formatted input or require impossibly large memory allocations. Consequently, the three defined errors will be the only ones that can appear legally in your output.

Note that the ppm formatted picture is likely to contain many unprintable characters, and there is no requirement that the message consists solely of printable characters either. The sample input printed on this page has '␣' to indicate spaces and bullets (•) that represent `0x0a` (linefeed). Each line is broken after the occurrence of `0x0a` for readability. To view your input and output, the *od* command should prove useful. To view the pictures as pictures, you can use *xloadimage*. A more interesting example is available via the *getdata* command: the picture, jellies.ppm, hides another picture, the GIF file, redrose.gif.

**Note** the space characters '␣' and `0x0a` '`<ffl>`' (•).

## Sample Input

```
P6•
8␣12•
255•
!}~␣~p␣}`␣~P␣~@~␣0␣~␣␣~•
0~~0~p0~`0~P0~@0~0/~!0~•
)}~@~p@~`@~P@~@@~0@~␣@~•
P~~P~pP~`P~PP~@P~0P~␣P~•
`~~`~p`~``~Q`~@`~0`~!`~•
o}}o~qp~`p~0p~@p~/p~!p~•
~q~~pp~p`~pP~p@~p1~p␣~p•
~`~~ap~``~_P~`@~`0~`␣~`•
~P~~Pp~P`~PP~P)}Q1~0_~Q•
~A~~@p~@a~@P~@@~@0~@␣~@•
~0~~0p~0`~0P~0@~00~0␣~0•
~␣~~␣p~␣`~␣P~␣@~␣0~␣␣~"•
```

## Sample Output

```
Hi!
```