Z85rot

Submit Solution

A software house wants to be able to transfer color images as text or to insert them directly into their code (again as textual data). To this aim they decide to use the Z85 encoding, that is a well-defined base 85 data representation that does not use characters classically reserved for strings delimiters or escape codes.

However, wanting to provide an (unsafe) obfuscation mechanism, they decide to apply an encrypting mechanism which consists in rotating the 85 symbols by a certain number of positions `N` at each output of a symbol. Referring to the Z85 standard, for example if `N = 13`, the value 27 is encoded with the symbol `'r'` if it is the first to be generated, but it is encoded with `'e'` if it is the second, with `'1'` if it is the third, with `'<'` if it is the fourth, and so on.

Create a command line program that accepts the following syntax:

```
Z85rot {c | d} <N> <input file> <output file>
```

The program, when executed with the first parameter equal to 'c' command accepts as input a file in PPM format and outputs a text file formatted as

```
<width in ASCII base ten>
<comma>
<height in ASCII base ten>
<comma>
<image pixels encoded with Z85 and rotations of N>
```

(there is no new line! It's just for visualization purposes)

Since the number of bytes to be encoded with Z85 must be a multiple of 4, if not the input is padded with bytes equal to zero.

Let's consider a 2x2 image with all red pixels:

255,0,0	255,0,0
255,0,0	255,0,0

In memory this is (hexadecimal):

```
FF 00 00 FF 00 00 FF 00 00 FF 00 00
```

The first value to be encoded is FF0000FF, that is 4.278.190.335, in base 85 is 81,81,27,6,0, which with Z85 (without rotations) would become `@@r60`, with a rotation of 1 it would become `@}p3@`.

The second value to be encoded is 00 00 FF 00, that is 65.280, in base 85 is 0,0,9,3,0, which with Z85 (without rotations) would become `00930`, with a rotation of 1 (continuing from the previous one) it would become `}{2})`.

The third value to be encoded is 00 FF 00 00, that is 16.711.680, in base 85 is 0,27,18,3,0, which with Z85 (without rotations) would become `0ri30`, with a rotation of 1 (continuing from the previous one) it would become `(g6(?`.

The image can thus be encoded as

```
2,2,@@r60009300ri30
```

without rotations, and as

```
2,2,@}p3@}{2})(g6(?
```

with a rotation of 1.