

# Show Me The Fax

Fax machines use a form of compression based on run-length encoding. Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, relatively simple graphic images such as icons, text, and line drawings. It is not useful with files that don't have many runs as it could potentially double the file size (photograph, for example).

For this problem, you will write a program that decodes a block of data using a very simple RLE algorithm. A run is encoded using a two byte sequence. The first byte of the sequence contains the count, and the second contains the value to repeat. The count is encoded using an 8 bit value with the high order bit set to 1. The remaining 7 bits represent the count-3. This gives a maximum run count of 130 per 2 byte sequence. (This implies that the minimum run count is 3). Bytes that are not part of a run are encoded as-is with a prefix byte indicating the count of bytes in the non-run minus 1, 0 through 127, representing a range of 1 - 128 (the high order bit will always be 0 in the case of non-run data).

## Input

The first line of input contains a single integer  $P$ , ( $1 \leq P \leq 1000$ ), which is the number of data sets that follow. Each data set consists of multiple lines. The first line contains two (2) decimal integer values: the problem number, followed by a space, followed by the number of bytes  $B$ , ( $1 \leq B \leq 5000$ ), to decode. The remaining line(s) contain(s) the data to be decoded. Each line of data to decode contains 80 hexadecimal digits (except the last line, which may contain less). 2 hexadecimal digits are used to represent each byte. Hexadecimal digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

## Output

For each data set, there are multiple lines of output. The first line contains a decimal integer giving the data set number followed by a single space, followed by a decimal integer giving the total number of decoded bytes. The remaining lines contain the decoded data, 80 hexadecimal digits per line, except the last line which may contain less.

## Example

### Input:

```
4
1 2
0007
2 4
00F481A5
3 32
850080FF016666825A0A717273747580080110111384550301020399807700CC
4 2
A568
```

### Output:

