MIDI Olympiad in Informatics 2007
Vilnius University, Lithuania

Input file	Output file	Time limit	Memory limit
`Utf-8.in`	`utf-8.out`	1 second	16 MB

UTF-8

For a long time, computer systems used limited character sets. It began as simple as ASCII, a 7-bit encoding covering most common characters used in Western languages.

With the rise of proprietary systems, various DOS and ANSI character sets were introduced. That lead to the creation of ISO-8859-* series. In the other parts of the world, specific encodings were created to facilitate efficient text storage.

The help came in the name of Unicode, which introduced a universal character set, defined by thousands of abstract characters, each identified by integer number, called code point. However, since each code point was considered to be 32 bits in size, it was rather inefficient to store texts as raw 4-byte words.

While looking for a good variable length encoding, Ken Thompson and Rob Pike of Bell Labs came up with UTF-8. It is based on idea that lower code points shall be encoded in no more bytes than higher ones. So, for example, we shall devote no more bytes to “A” (Unicode 0x0041) than to “♫” (Unicode 0x266B). Each Unicode character has unique representation in UTF-8. The following table explains the technique in more detail.

Code point range (hexadecimal)	**Encoded (UTF-8) – in binary**	**Bytes (UTF-8)**
000000–00007F	`0zzzzzzz`	1
000080–0007FF	`110yyyyy 10zzzzzz`	2
000800–00FFFF	`1110xxxx 10yyyyyy 10zzzzzz`	3
010000–10FFFF	`11110www 10xxxxxx 10yyyyyy 10zzzzzz`	4

For example, the character aleph (א), which has Unicode code point 0x05D0, is encoded into UTF-8 in this way:

- It falls into the range of 0x0080–0x07FF. The table shows it will be encoded using two bytes, `110yyyyy 10zzzzzz`.
- Hexadecimal 0x05D0 is equivalent to binary `10111010000`.
- The eleven bits are put in their order into the positions marked by "y"-s and "z"-s: `110`**`10111`** `10`**`010000`**.
- The final result is the two bytes, more conveniently expressed as the two hexadecimal bytes `0xD7 0x90`. That is the encoding of the character aleph (א) in UTF-8.

You are given a file that is supposedly encoded with UTF-8 encoding. Your task is to check the given file and report if it is indeed correctly encoded. Example input and output is presented below.

- In case the file is correctly encoded, you shall output “OK” followed by space and number of characters decoded in the file given.
- Otherwise you shall output “FAIL” followed by space and the number of characters decoded just before encountering malformed bytes.