2001/2002 ACM SOUTHERN CALIFORNIA REGIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 6
Sync Code

As part of your graduate research work at Swamp County College, your advisor has requested that you analyze sets of remote-sensing data collected from a satellite. Processed data from this particular satellite is publicly available, but only at a premium. However, the raw, unprocessed data is for sale at minimal cost. Your advisor has decided to give you the opportunity to exercise your programming skills on the raw data.

You have been provided with documentation of the raw data format and with files that (supposedly) contain the raw data. As a first step, you will verify the presence of data within the files. The data within each file has been stored as it was received from the satellite, in a single bit-stream. Within a file, one or multiple segments of valid data may exist, separated by segments where no data was collected or by segments of garbled (noisy) data. No data or garbled data may also occur at the beginning and/or end of a file. In the worst case, a file may contain no valid data.

The raw data is partitioned into packets of 80 bytes each. Bytes are numbered starting at 0. Assume 8 bits/byte. Bytes 4–7 of each packet contain a distinct 32-bit pattern known as the synchronization, or *sync* code. The sync code, which is a constant for the satellite, acts as a marker by which the presence of the raw data can be recognized within the bit stream. Bytes 9–12 of each packet contain a counter, represented as a 32-bit unsigned integer. Note, however, that no byte alignment was performed on the data before delivery to you. As a result, the bytes of the data packets may be bit-shifted relative to the byte boundaries of the data stored in each file. You may assume that the relative offsets between packet byte boundaries and data byte boundaries are constant within a given file.

A valid packet is defined to be a complete packet containing the sync code in the correct location. For the purposes of this discussion, an invalid packet is an 80-byte length that does not contain the sync code in the correct location. Since the sync code can occur naturally anywhere in the bit stream, the beginning of a valid data segment is (arbitrarily) defined to be three or more contiguous valid packets. Once the beginning of a valid data segment has been found, the end of the segment is defined to be the end of the last valid packet before three or more invalid packets OR the end of the last valid packet before end-of-file, whichever comes first. If a valid data segment is followed by fewer than three invalid packets, you should treat those invalid packets as valid packets (and thus as part of the valid data segment).

Given an input file, your program is to locate all segments of valid data within the file and report for each segment

1. The offset, relative to the start of the file, of the first bit of the segment, where bit offsets are numbered starting at 0. Express this value as a non-negative integer.
2. The number of packets in the segment. Express this value as a positive integer.
3. The value of the counter for the first packet in the segment. Express this value as a non-negative integer.

An input file will be a binary file of nonzero length up to a maximum of 1 MB. For the sync code and counter fields, bytes are ordered from high to low. Bits are ordered from high to low within a byte. The satellite's sync code is (hexadecimal) 03915ED3.

For each segment, report the offset, number of packets, and counter in that order on a single line. All values should be expressed in decimal. Use a single space to separate each of the values. If your program finds no valid data in the file, it should print the single line '`No valid data found`' terminated by a newline.

Problem 6
Sync Code (continued)

Note: Input to your program will be a binary file. For readability, the sample input has been expressed in hexadecimal format with whitespace added after each byte.

Sample Input

```
08 01 01 01 00 39 15 ED 30 00 00 00 60 00 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 00 39 15 ED 30 00 00 00 60 10 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 00 39 15 ED 30 00 00 00 60 20 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
```

Sample Output

```
4 3 1536
```