2004/2005 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
Temperature Data Compression

Dirt Cheap Electronics (DCE), a leading manufacturer of electronics for budget-conscious companies, has decided to produce a small, battery operated temperature recording and logging device. The value of these devices is that they can be recovered after deployment, and the entire log of temperatures can be downloaded to a computer for later analysis. The device will read temperatures in steps of whole degrees C, in what is known as the extended industrial temperature range, -40 to 125 degrees C. Your team must compress the temperature readings that will be stored onboard the device.

The microcontroller unit (μCU) that DCE plans to use in the prototype has only 128 bytes of EEPROM for long term storage (logging). To store a significant number of temperature readings, it must compress the logged temperatures. The engineering staff took very expensive thermal metering and logging computers into the intended environments of the new device. After analyzing the data, the staff discovered that recording an initial temperature, then encoding only the change (delta) between consecutive readings promised good compression. The engineers then specified an encoding technique, and handed off all the logged test data and the specification to the software manager for implementation.

Compressed temperature data is a series of records with varying bit lengths. Multi-bit values are encoded in big-endian bit order. Each record begins with a data presence bit:

0: When the data presence bit is 0, no data follows (this marks the end of the bit stream).

1: When the data presence bit is 1, the header consists of an additional seven bits of the form **`issnnnn`**:

`i` The second bit is 1 when an initial 8-bit two's complement value follows the header, and is 0 when only temperature deltas follow the header.

`ss` The following two bits of the data header correspond to the size of the coded deltas: **`00`** means that each delta reading in the record is a 2-bit two's complement integer, [-2..1]. **`01`** means that each delta is a 3-bit two's complement integer, [-4..3]. **`10`** means that each delta is a 4-bit two's complement integer, [-8..7]. **`11`** means that each delta is a 5-bit two's complement integer, [-16..15].

`nnnn` The last four bits form an unsigned count (measurements - 1) of how many temperature values (optional initial value plus deltas) are present in the record: thus, 0 indicates that only one measurement follows the header, and 15 indicates that 16 measurements follow the header.

The software manager recognized that the limited RAM in the μCU for the compression algorithm prevents optimal compression for an entire set of measurements. The readings will be subject to a compression window of 16 measurements. The software manager has tasked your team to produce an algorithm to compress a window's worth (0..16 measurements) of data. Your program must read a series of windowed data from multiple independent tests. For each window, form one or more records that compress the window optimally (using the fewest bits), and display the resulting bit stream.

Input is a series of test windows. Each window is a sequence of temperature readings in the range [-40..125], one per line. The end of a test window is signaled by a dummy temperature value of -128. The end of the series is signaled by end-of-file.

Output is one line per test recording. Each line consists of a string of zeros and ones, representing the compressed data. There should be no spaces anywhere in the output lines. The bits must represent an optimal compression.

Problem 3
Temperature Data Compression (continued)

Sample Input

```
25
26
27
37
38
38
20
19
17
-128
-128
```

Output for the Sample Input

```
1100001000011001010111000010001001010100110000100001010011100
0
```