F • I²C

I²C (**Inter-Integrated Circuit**) is a serial communication protocol that is used to attach low-speed peripherals (~100 *kbit/sec*) to a motherboard, embedded system or cell phone. A single I²C data bus may have several devices attached, each with a different 7-bit *address.* One of the nice things about I²C is that it only requires two signal lines, SCL (clock) and SDA (data). One bit of data is presented on the I²C data bus (SDA line) per clock (SCL). Typically, one device on the bus is designated as the *master,* and the other devices are *slaves.* The *master* will initiate communication to a specific device on the bus by specifying its *address* in a transaction.

If there is no activity on the I²C bus, both the SCL and SDA signals are in a *high* state (1). The master initiates a transaction on the bus by pulling the SDA signal to a *low* state (0), while the SCL signal is *high* (1): this is called a **START** bit. At this point, all slaves on the bus must start paying attention to the signaling to see if the transaction is directed at them. The *master* will then send the 7-bit slave address (most significant bit first), one bit at-a-time. This is done by bringing the SCL signal low (0), presenting the next bit value on the SDA line, then releasing the SCL signal so it goes high (1). The slaves will read the SDA signal as soon as the clock goes high (1). This operation is repeated 7 times, one for each bit of the desired slave address. Another data bit is presented on the bus in the same manner. This last bit is an indicator as to whether the master wants to read from (1) or write to (0) the addressed slave device. When a slave recognizes its address on the bus, it must acknowledge (*ACK*) that it is available and ready by pulling the SDA line low. The master will see this the next time it brings the clock high, at which point, the data transfer can begin. If no *ACK* is seen this means that the slave specified by the address does not exist. Note: If no device pulls a signal low, it will go high by default; a device simply *releases* a signal, and it will go high.

Data is always transferred as 8 bit bytes, 1 bit at-a-time, most significant bit first. After each byte, the slave must *ACK* the master by pulling the SDA line low. If the slave is not ready to transmit (or receive) the next byte of data, it may pull the SCL line low. This will cause the master to go into a *wait mode* until the slave is ready. The slave indicates it is ready by bringing SDA low, and releasing the SCL line so it goes high. The next byte of data can then be transferred. The sequence repeats until the master decides all the data has been transferred, at which point it will send a **STOP** bit. This is done when the master lets the SDA line go high while the SCL line is high.

For this problem, you will write a program that *sniffs* the I²C bus signals and displays the details of transactions.

Input

The first line of input contains a single integer P, ($1 \le P \le 1000$), which is the number of data sets that follow. Each data set consists of multiple lines which represents a single I²C transaction. The first line contains two (2) decimal integer values: the problem number, followed by a space, followed by the number of signal samples S, ($1 \le S \le 1161$), for the transaction. The remaining line(s) contain(s) the signal samples. Each line of samples contains 40 samples (except the last which may contain less). Each sample consists of 2 binary digits characters representing SCL and SDA in that order.

Output

For each data set, display a single line containing a decimal integer giving the data set number followed by a single space, followed by a description of the transaction. There will only be six different descriptions (two non-error cases, and four error cases):

Non-error cases:

```
WRITE OF n BYTES TO SLAVE xx
READ OF n BYTES FROM SLAVE xx
```

Error cases:

```
ERROR NO START BIT
ERROR NO STOP BIT
ERROR NO ACK FROM SLAVE xx
ERROR NO ACK FOR DATA
```

n is a decimal integer (1 – 128) representing the number of data bytes.
xx is a 2 digit hexadecimal value (00-7F) representing the slave address.
The **`ERROR NO ACK FROM SLAVE xx`** case occurs when there is no **ACK** for the supplied address
The **`ERROR NO ACK FOR DATA`** case occurs when there is no **ACK** after a data byte

For the error cases, only the *first* error detected should be displayed.

Sample Input

```
4
1 97
01111001110010001000100111011101110111001000100010011100100010001000100010001000
10001001110010001000100010011100100010001001110010001000100111001000100010001001
1100100010001001110111001000101111
2 169
01111000100010011100100010001001110010001000100010011100100010001000100010001000
10001001110010001000100010011100100010001001110010001000100111001000100010001001
11001000100010011101110010001000100111001000100111001000100010001000100111001000
10011100100111001000100010011100100010011101110010001000100010011100100010011101
110111001000101111
3 60
01111000100010001001110010011101110010001000100010011100100010001000100010001000
1000100111001000100010001001110010001111
4 40
0111100010001001110111001001110010011100111111111111111111111111111111111111111
```

Sample Output

```
1 READ OF 4 BYTES FROM SLAVE 47
2 WRITE OF 8 BYTES TO SLAVE 11
3 ERROR NO STOP BIT
4 ERROR NO ACK FROM SLAVE 0B
```