Problem A: The Errant Physicist

The well-known physicist Alfred E Neuman is working on problems that involve multiplying polynomials of x and y. For example, he may need to calculate

$$(-x^8y + 9x^3 - 1 + y).(x^5y + 1 + x^3)$$

getting the answer

$$-x^{13}y^2 - x^{11}y + 8x^8y + 9x^6 - x^5y + x^5y^2 + 8x^3 + x^3y - 1 + y$$

Unfortunately, such problems are so trivial that the great man's mind keeps drifting off the job, and he gets the wrong answers. As a consequence, several nuclear warheads that he has designed have detonated prematurely, wiping out five major cities and a couple of rain forests.

You are to write a program to perform such multiplications and save the world.

The file of input data will contain pairs of lines, with each line containing no more than 80 characters. The final line of the input file contains a # as its first character. Each input line contains a polynomial written without spaces and without any explicit exponentiation operator. Exponents are positive non-zero unsigned integers. Coefficients are also integers, but may be negative. Both exponents and coefficients are less than or equal to 100 in magnitude. Each term contains at most one factor in x and one in y. For example, an input file for the above problem might be:

```
-yx8+9x3-1+y
x5y+1+x3
1
1
#
```

Your program must multiply each pair of polynomials in the input, and print each product on a pair of lines, the first line containing all the exponents, suitably positioned with respect to the rest of the information, which is in the line below. For example, the output for the above input would be

```
  13 2     11        8        6     5       5 2      3       3
-x   y   - x   y + 8x y + 9x   - x y + x y   + 8x   + x y - 1 + y

1
```

The following rules control the output format:

1. Terms in the output line must be sorted in decreasing order of powers of x and, for a given power of x, in increasing order of powers of y.

2. Like terms must be combined into a single term. For example, $42x^2y^3 - 40x^2y^3$ is replaced by $2x^2y^3$.

3. Terms with a zero coefficient must not be displayed.

4. Coefficients of 1 are omitted, except for the case of a constant term of 1.

5. Exponents of 1 are omitted.

6. Factors of x^0 and y^0 are omitted.

7. Binary pluses and minuses (that is the pluses and minuses connecting terms in the output) have a single blank column both before and after.

8. If the coefficient of the first term is negative, it is preceded by a unary minus in the first column, with no intervening blank column. Otherwise, the coefficient itself begins in the first output column.

9. The output can be assumed to fit into a single line of at most 80 charactes in length.

10. There should be no blank lines printed between each pair of output lines.

11. The pair of lines that contain a product should be the same length—trailing blanks should appear after the last non-blank character of the shorter line to achieve this.

The above example conforms to all those requirements.

Problem B: "Accordian" Patience

You are to simulate the playing of games of "Accordian" patience, the rules for which are as follows:

> Deal cards one by one in a row from left to right, not overlapping. Whenever the card matches its immediate neighbour on the left, or matches the third card to the left, **it** may be moved onto that card. Cards match if they are of the same suit or same rank. After making a move, look to see if it has made additional moves possible. Only the top card of each pile may be moved at any given time. Gaps between piles should be closed up as soon as they appear by moving all piles on the right of the gap one position to the left. Deal out the whole pack, combining cards towards the left whenever possible. The game is won if the pack is reduced to a single pile.

Situations can arise where more than one play is possible. Where two cards may be moved, you should adopt the strategy of always moving the leftmost card possible. Where a card may be moved either one position to the left or three positions to the left, move it three positions.

Input data to the program specifies the order in which cards are dealt from the pack. The input contains pairs of lines, each line containing 26 cards separated by single space characters. The final line of the input file contains a # as its first character. Cards are represented as a two character code. The first character is the face-value (A=Ace, 2–9, T=10, J=Jack, Q=Queen, K=King) and the second character is the suit (C=Clubs, D=Diamonds, H=Hearts, S=Spades).

One line of output must be produced for each pair of lines (that between them describe a pack of 52 cards) in the input. Each line of output shows the number of cards in each of the piles remaining after playing "Accordian patience" with the pack of cards as described by the corresponding pairs of input lines. For example, given the input

```
QD AD 8H 5S 3H 5H TC 4D JH KS 6H 8S JS AC AS 8D 2H QS TS 3S AH 4H TH TD 3C 6S
8C 7D 4C 4S 7S 9H 7C 5D 2S KD 2D QH JD 6D 9D JC 2C KH 3D QC 6C 9S KC 7H 9C 5C
#
```

the output would be:

```
6 piles remaining: 40 8 1 1 1 1
```

Problem C: Software CRC

You work for a company which uses lots of personal computers. Your boss, Dr Penny Pincher, has wanted to link the computers together for some time but has been unwilling to spend any money on the Ethernet boards you have recommended. You, unwittingly, have pointed out that each of the PCs has come from the vendor with an asynchronous serial port at no extra cost. Dr Pincher, of course, recognizes her opportunity and assigns you the task of writing the software necessary to allow communication between PCs.

You've read a bit about communications and know that every transmission is subject to error and that the typical solution to this problem is to append some error checking information to the end of each message. This information allows the receiving program to detect when a transmission error has occurred (in most cases). So, off you go to the library, borrow the biggest book on communications you can find and spend your weekend (unpaid overtime) reading about error checking.

Finally you decide that CRC (cyclic redundancy check) is the best error checking for your situation and write a note to Dr Pincher detailing the proposed error checking mechanism noted below.

CRC Generation

The message to be transmitted is viewed as a long positive binary number. The first byte of the message is treated as the most significant byte of the binary number. The second byte is the next most significant, etc. This binary number will be called "m" (for message). Instead of transmitting "m" you will transmit a message, "m2", consisting of "m" followed by a two-byte CRC value.

The CRC value is chosen so that "m2" when divided by a certain 16-bit value "g" leaves a remainder of 0. This makes it easy for the receiving program to determine whether the message has been corrupted by transmission errors. It simply divides any message received by "g". If the remainder of the division is zero, it is assumed that no error has occurred.

You notice that most of the suggested values of "g" in the book are odd, but don't see any other similarities, so you select the value 34943 for "g" (the generator value).

You are to devise an algorithm for calculating the CRC value corresponding to any message that might be sent. To test this algorithm you will write a program which reads lines (each line being all characters up to, but not including the end of line character) as input, and for each line calculates the CRC value for the message contained in the line, and writes the numeric value of the CRC bytes (in hexadecimal notation) on an output line. Each input line will contain no more than 1024 ASCII characters. The input is terminated by a line that contains a # in column 1.

For example, if the input consists of:

```
this is a test<newline>
A<newline>
#<newline>
```

you will calculate the CRC values for the 14 byte message:

```
this is a test
```

and the 1 byte message:

```
A
```

and the corresponding output should be:

```
77 FD
0C 86
```

Note: each CRC printed should be in the range 0 to 34942 (decimal).

Problem D: Krypton Factor

You have been employed by the organisers of a Super Krypton Factor Contest in which contestants have very high mental and physical abilities. In one section of the contest the contestants are tested on their ability to recall a sequenace of characters which has been read to them by the Quiz Master. Many of the contestants are very good at recognising patterns. Therefore, in order to add some difficulty to this test, the organisers have decided that sequences containing certain types of repeated subsequences should not be used. However, they do not wish to remove all subsequences that are repeated, since in that case no single character could be repeated. This in itself would make the problem too easy for the contestants. Instead it is decided to eliminate all sequences containing an occurrence of two adjoining identical subsequences. Sequences containing such an occurrence will be called "easy". Other sequences will be called "hard".

For example, the sequence ABACBCBAD is easy, since it contains an adjoining repetition of the subsequence CB. Other examples of easy sequences are:

- BB

- ABCDACABCAB

- ABCDABCD

Some examples of hard sequences are:

- D

- DC

- ABDAB

- CBABCBA

In order to provide the Quiz Master with a potentially unlimited source of questions you are asked to write a program that will read input lines that contain integers n and L (in that order), where $n > 0$ and L is in the range $1 \leq L \leq 26$, and for each input line prints out the nth hard sequence (composed of letters drawn from the first L letters in the alphabet), in increasing alphabetical order[1], followed (on the next line) by the length of that sequence. The first sequence in this ordering is A. You may assume that for given n and L there do exist at least n hard sequences.

For example, with $L = 3$, the first 7 hard sequences are:

```
A
AB
ABA
ABAC
ABACA
ABACAB
ABACABA
```

As each sequence is potentially very long, split it into groups of four (4) characters separated by a space. If there are more than 15 such groups, please start a new line for the 16th group.

Therefore, if the integers 7 and 3 appear on an input line, the output lines produced should be

---

[1]Alphabetical ordering here corresponds to the normal ordering encountered in a dictionary.

```
ABAC ABA
7
```

As a second example, if the integers 30 and 3 appear on an input line, then the output lines produced should be

```
ABAC ABCA CBAB CABA CABC ACBA CABA
28
```

Input is terminated by a line containing two zeroes. Your program may assume a maximum sequence length of 80. It must produce its answer within 10 seconds.

Problem E: Roman Roulette

The historian Flavius Josephus relates how, in the Romano-Jewish conflict of 67 A.D., the Romans took the town of Jotapata which he was commanding. Escaping, Jospehus found himself trapped in a cave with 40 companions. The Romans discovered his whereabouts and invited him to surrender, but his companions refused to allow him to do so. He therefore suggested that they kill each other, one by one, the order to be decided by lot. Tradition has it that the means for effecting the lot was to stand in a circle, and, beginning at some point, count round, every third person being killed in turn. The sole survivor of this process was Josephus, who then surrendered to the Romans. Which begs the question: had Josephus previously practised quietly with 41 stones in a dark corner, or had he calculated mathematically that he should adopt the 31st position in order to survive?

Having read an account of this gruesome event you become obsessed with the fear that you will find yourself in a similar situation at some time in the future. In order to prepare yourself for such an eventuality you decide to write a program to run on your hand-held PC which will determine the position that the counting process should start in order to ensure that you will be the sole survivor.

In particular, your program should be able to handle the following variation of the processes described by Josephus. $n > 0$ people are initially arranged in a circle, facing inwards, and numbered from 1 to n. The numbering from 1 to n proceeds consecutively in a clockwise direction. Your allocated number is 1. Starting with person number i, counting starts in a clockwise direction, until we get to person number k $(k > 0)$, who is promptly killed. We then proceed to count a further k people in a clockwise direction, starting with the person immediately to the left of the victim. The person number k so selected has the job of burying the victim, and then returning to the position in the circle that the victim had previously occupied. Counting then proceeeds from the person to his immediate left, with the kth person being killed, and so on, until only one person remains.

For example, when $n = 5$, and $k = 2$, and $i = 1$, the order of execution is 2, 5, 3, and 1. The survivor is 4.

Your program must read input lines containing values for n and k (in that order), and for each input line output the number of the person with which the counting should begin in order to ensure that you are the sole survivor. For example, in the above case the safe starting position is 3. Input will be terminated by a line containing values of 0 for n and k.

Your program may assume a maximum of 100 people taking part in this event.

Problem F: The Psychic Poker Player

In 5-card draw poker, a player is dealt a hand of five cards (which may be looked at). The player may then discard between zero and five of his or her cards and have them replaced by the same number of cards from the top of the deck (which is face down). The object is to maximize the value of the final hand. The different values of hands in poker are given at the end of this problem.

Normally the player cannot see the cards in the deck and so must use probability to decide which cards to discard. In this problem, we imagine that the poker player is psychic and knows which cards are on top of the deck. Write a program which advises the player which cards to discard so as to maximize the value of the resulting hand.

Input will consist of a series of lines, each containing the initial five cards in the hand then the first five cards on top of the deck. Each card is represented as a two-character code. The first character is the face-value (A=Ace, 2–9, T=10, J=Jack, Q=Queen, K=King) and the second character is the suit (C=Clubs, D=Diamonds, H=Hearts, S=Spades). Cards will be separated by single spaces. Each input line will be from a single valid deck, that is there will be no duplicate cards in each hand and deck.

Each line of input should produce one line of output, consisting of the initial hand, the top five cards on the deck, and the best value of hand that is possible. Input is terminated by end of file.

Use the sample input and output as a guide.

Sample Input

```
TH JH QC QD QS QH KH AH 2S 6S
2H 2S 3H 3S 3C 2D 3D 6C 9C TH
2H 2S 3H 3S 3C 2D 9C 3D 6C TH
2H AD 5H AC 7H AH 6H 9H 4H 3C
AC 2D 9C 3S KD 5S 4D KS AS 4C
KS AH 2H 3C 4H KC 2C TC 2D AS
AH 2C 9S AD 3C QH KS JS JD KD
6C 9C 8C 2D 7C 2H TC 4C 9S AH
3D 5S 2H QD TD 6S KH 9H AD QH
```

Sample Output

```
Hand: TH JH QC QD QS Deck: QH KH AH 2S 6S Best hand: straight-flush
Hand: 2H 2S 3H 3S 3C Deck: 2D 3D 6C 9C TH Best hand: four-of-a-kind
Hand: 2H 2S 3H 3S 3C Deck: 2D 9C 3D 6C TH Best hand: full-house
Hand: 2H AD 5H AC 7H Deck: AH 6H 9H 4H 3C Best hand: flush
Hand: AC 2D 9C 3S KD Deck: 5S 4D KS AS 4C Best hand: straight
Hand: KS AH 2H 3C 4H Deck: KC 2C TC 2D AS Best hand: three-of-a-kind
Hand: AH 2C 9S AD 3C Deck: QH KS JS JD KD Best hand: two-pairs
Hand: 6C 9C 8C 2D 7C Deck: 2H TC 4C 9S AH Best hand: one-pair
Hand: 3D 5S 2H QD TD Deck: 6S KH 9H AD QH Best hand: highest-card
```

Note that the order of the cards in the player's hand is irrelevant, but the order of the cards in the deck is important because the discarded cards must be replaced from the top of the deck. Also note that examples of all types of hands appear in the sample output, with the hands shown in decreasing order of value.