Problem S: Runs and Flushes

Write a program that will read in a representation of a deck of cards and then determine the longest flush and the longest run present in the deck. A *flush* is a series of cards all of the same suit and a *run* is a series of cards that follow one another in *ascending* numerical sequence, with Ace following King and preceding Two. Thus ♠2 ♠5 ♠K ♠3 ♠A forms a flush of length 5, ♣9 ♢10 ♣J ♠Q ♡K ♣A ♢2 forms a run of length 7 and ♡2 ♡3 ♡4 ♡5 ♡6 forms both a flush and a run of length 5.

Input will consist of a series of decks of cards, each deck occupying two lines. The input will be terminated by a line consisting of a single #. Each card will be represented by a two character string—the suit (S, H, D, C) and the value (A for Ace, 2–9 for two through nine, T for ten, J for Jack, Q for Queen and K for King). There will be 26 cards per line.

Output will consist of one line for each deck and will contain the length of the longest flush followed by the length of the longest run, each right justified in a field of width 6.

Sample input

```
CQDTC4D8S7HTDAH7D2S3D6C6S6D9S4SAD7H2CKH5D3CTS8C9H3C3
DQS9SQDJH8HAS2SKD4H4S5C7SJC8DKC5C2CAHQCJSTH6HKH9D5HJ
#
```

Sample output

```
     3     2
```

Problem T: Family Concepts

An interesting parlour game is known as 'Family Concepts'. One person thinks of a concept that defines a class of word and then describes a family as liking instances of the concept and not liking non-instances of the concept. When a member of the party guesses the concept, she can assist in describing this mythical family. As more and more participants guess the concept they too enter the game, and the descriptions usually get more and more outrageous, until every one has guessed the concept and joined in the fun.

A typical concept is 'the word contains a double letter' in which case the family would:

> like sheep and cattle but neither pigs nor horses,
> eat sweets but not chocolates,
> live in a village, but not in a town or city
> etc.

Write a program that will determine whether a given word fits this concept, that is has a double letter, or not.

Input will consist of a series of lines, each containing a word consisting of up to 20 lower case letters only. The file will be terminated by a line consisting of a single #.

Output will consist of a series of lines, one for each line in the input file. Each line will consist of the word starting in column 1 and then, ending in column 24, one of the words 'yes' or 'no' depending on whether the word has at least two adjacent letters the same.

Sample input

```
street
village
chocolates
abcdefghijklmnopqrst
#
```

Sample output

```
street               yes
village              yes
chocolates            no
abcdefghijklmnopqrst  no
```

Problem U: Russell Soundex Coding

In many situations it is desirable that surnames that sound similar should occur together, for instance all the variants of Smith (Smit, Smythe) should appear together in a telephone directory. One system of encoding names to assist in this is known as the Russell Soundex encoding.

The rules for encoding are as follows:

1. The code consists of the initial letter of the surname followed by 3 digits derived from subsequent letters in the name.
2. The letters A, E, H, I, O, U, W, Y are pseudo-vowels and do not form part of the code—the remaining letters are called codable letters. Hyphens, quotes, etc are ignored.
3. Initially all codable letters in the surname (including the first) are encoded according to the following table, although they may not all be used.

1	B, F, P, V
2	C, G, J, K, S, X, Z
3	D, T
4	L
5	M, N
6	R

4. The encoding is then scanned for sequences of repeated digits. All but the first of these are discarded unless the letters they were derived from are separated by a pseudo-vowel.
5. The first three remaining digits are used as the code. If there are less than three digits, the code is padded with zeroes to the right.

Some examples follow to illustrate the workings of the various rules.

```
IRVINE     =  I R V I N E
              I 6 1 * 5 * = I615          coding
              1 3 3 2 3 2                 rules used

SMITHSON   =  S M I T H S O N
              S 5 * 3 * 2 * * = S532      coding
              1 3 2 3 2 3 2 5             rules used

LLEWELLYN  =  L L E W E L L Y N
              L * * * * 4 * * 5 = L450    coding
              1 4 2 2 2 3 4 2 2 (5)       rules used
```

Write a program that will accept surnames and determine their codes.

Input will consist of a series of surnames, one per line. Each surname will be a string containing no more than 20 characters. All letters will be in upper case. The input will be terminated by a line consisting of a single #.

Output will consist of a series of lines containing the given surname, left justified in a field of 21 characters, and the encoding for that name.

Sample input

```
IRVINE
LLEWELLYN
#
```

Sample output

```
IRVINE                   I615
LLEWELLYN                L450
```

Problem V: Caesar Cypher

One of the earliest encrypting systems is attributed to Julius Caesar: if the letter to be encrypted is the Nth letter in the alphabet, replace it with the (N+K)th where K is some fixed integer (Caesar used K = 3). We usually treat a space as zero and all arithmetic is then done modulo 27. Thus for K = 1 the message 'ATTACK AT DAWN' becomes 'BUUBDLABUAEBXO'.

Decrypting such a message is trivial since one only needs to try 26 different values of K. This process is aided by knowledge of the language, since then one can determine when the decrypted text forms recognisable words. If one does not know the language, then a dictionary would be necessary.

Write a program that will read in a dictionary and some encrypted text, determine the value of K that was used, and then decrypt the cyphertext to produce the original message. The original message contained only letters and spaces and has been encrypted using the above method. The most suitable value of K will be the one which produces the most matches with the words in the dictionary.

Input will consist of a dictionary and the encrypted text. The dictionary will consist of no more than 100 lines each containing a word in upper case characters and not more than 20 characters in length. The dictionary portion will be terminated by a line consisting of a single #. The encrypted text will follow immediately and will consist of a single line containing no more than 250 characters. Note that the dictionary will not necessarily contain all the words in the original text, although it will certainly contain a large proportion of them. It may also contain words that are not in the original text. The dictionary will not appear in any particular order.

Output will consist of the decrypted text. Lines should be as long as possible, but not exceeding 60 characters and no word may cross a line break.

Sample input

```
THIS
DAWN
THAT
THE
ZORRO
OTHER
AT
THING
#
BUUBDLA PSSPABUAEBXO
```

Sample output

```
ATTACK ZORRO AT DAWN
```

Problem W: Gondwanaland Telecom

Gondwanaland Telecom makes charges for calls according to distance and time of day. The basis of the charging is contained in the following schedule, where the charging step is related to the distance:

Charging Step (distance)	Day Rate 8am to 6pm	Evening Rate 6pm to 10pm	Night Rate 10pm to 8am
A	0.10	0.06	0.02
B	0.25	0.15	0.05
C	0.53	0.33	0.13
D	0.87	0.47	0.17
E	1.44	0.80	0.30

All charges are in dollars per minute of the call. Calls which straddle a rate boundary are charged according to the time spent in each section. Thus a call starting at 5:58 pm and terminating at 6:04 pm will be charged for 2 minutes at the day rate and for 4 minutes at the evening rate. Calls less than a minute are not recorded and no call may last more than 24 hours.

Write a program that reads call details and calculates the corresponding charges.

Input lines will consist of the charging step (upper case letter 'A'..'E'), the number called (a string of 7 digits and a hyphen in the approved format) and the start and end times of the call, all separated by exactly one blank. Times are recorded as hours and minutes in the 24 hour clock, separated by one blank and with two digits for each number. Input will be terminated by a line consisting of a single #.

Output will consist of the called number, the time in minutes the call spent in each of the charge categories, the charging step and the total cost in the format shown below.

Sample input

```
A 183-5724 17 58 18 04
#
```

Sample output

```
         10    16    22    28 31      39
  183-5724     2     4     0  A     0.44
```

Problem X: Student Grants

The Government of Impecunia has decided to discourage tertiary students by making the payments of tertiary grants a long and time-consuming process. Each student is issued a student ID card which has a magnetically encoded strip on the back which records the payment of the student grant. This is initially set to zero. The grant has been set at \$40 per year and is paid to the student on the working day nearest to his birthday. (Impecunian society is still somewhat medieval and only males continue with tertiary education.) Thus on any given working day up to 25 students will appear at the nearest office of the Department of Student Subsidies to collect their grant.

The grant is paid by an Automatic Teller Machine which is driven by a reprogrammed $8085\frac{1}{2}$ chip originally designed to run the state slot machine. The ATM was built in the State Workshops and is designed to be difficult to rob. It consists of an interior vault where it holds a large stock of \$1 coins and an output store from which these coins are dispensed. To limit possible losses it will only move coins from the vault to the output store when that is empty. When the machine is switched on in the morning, with an empty output store, it immediately moves 1 coin into the output store. When that has been dispensed it will then move 2 coins, then 3, and so on until it reaches some preset limit k. It then recycles back to 1, then 2 and so on.

The students form a queue at this machine and, in turn, each student inserts his card. The machine dispenses what it has in its output store and updates the amount paid to that student by writing the new total on the card. If the student has not received his full grant, he removes his card and rejoins the queue at the end. If the amount in the store plus what the student has already received comes to more than \$40, the machine only pays out enough to make the total up to \$40. Since this fact is recorded on the card, it is pointless for the student to continue queuing and he leaves. The amount remaining in the store is then available for the next student.

Write a program that will read in values of N (the number of students, $1 \leq N \leq 25$) and k (the limit for that machine, $1 \leq k \leq 40$) and calculate the order in which the students leave the queue.

Input will consist of a series of lines each containing a value for N and k as integers. The list will be terminated by two zeroes (0 0).

Output will consist of a line for each line of input and will contain the list of students in the order in which they leave the queue. Students are ordered according to their position in the queue at the start of the day. All numbers must be right justified in a field of width 3.

Sample input

```
5 3
0 0
```

Sample output

```
  1  3  5  2  4
```

Problem Y: Amicable Numbers

The factor sum of a number is the sum of all its factors, including 1 but excluding itself. Thus the factor sum of 8 is $1 + 2 + 4 = 7$, the factor sum of 25 is $1 + 5 = 6$. (Note that a factor of a number divides it exactly).

We define amicable pairs as numbers where each member of the pair is the factor sum of the other, e.g. 284 and 220.

Factor sum of 284: $1 + 2 + 4 + 71 + 142 = 220$
Factor sum of 220: $1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$

Write a program that will read in pairs of numbers and determine whether or not they form an amicable pair.

Input lines will consist of a pair of integers less than 10,000. Input will be terminated by a line containing a pair of zeroes (0 0).

Output will consist of a series of lines, one for each line of the input, containing the numbers right justified in a field of width 5 and, starting in column 12, one of the words 'yes' or 'no', depending on whether the numbers form an amicable pair or not.

Sample input

```
284 220
1280 1786
0 0
```

Sample output

```
  284  220 yes
 1280 1786 no
```

Problem Z: String Transformations

Write a program that will accept strings of up to 60 characters and transform them according to the following rules.

Transform all uppercase letters to the equivalent lower case characters, similarly transform all lower case characters to uppercase characters. All blanks are to be transformed to '#' and all other characters become '?'.

Input lines will consist of a string to be transformed. Input will be terminated by a line consisting of a single #.

Output will consist of the transformed strings, one per line.

Sample input

```
This, believe it (or not) is a String//!
SO IS THIS
#
```

Sample output

```
tHIS?#BELIEVE#IT#?OR#NOT?#IS#A#sTRING???
so#is#this
```