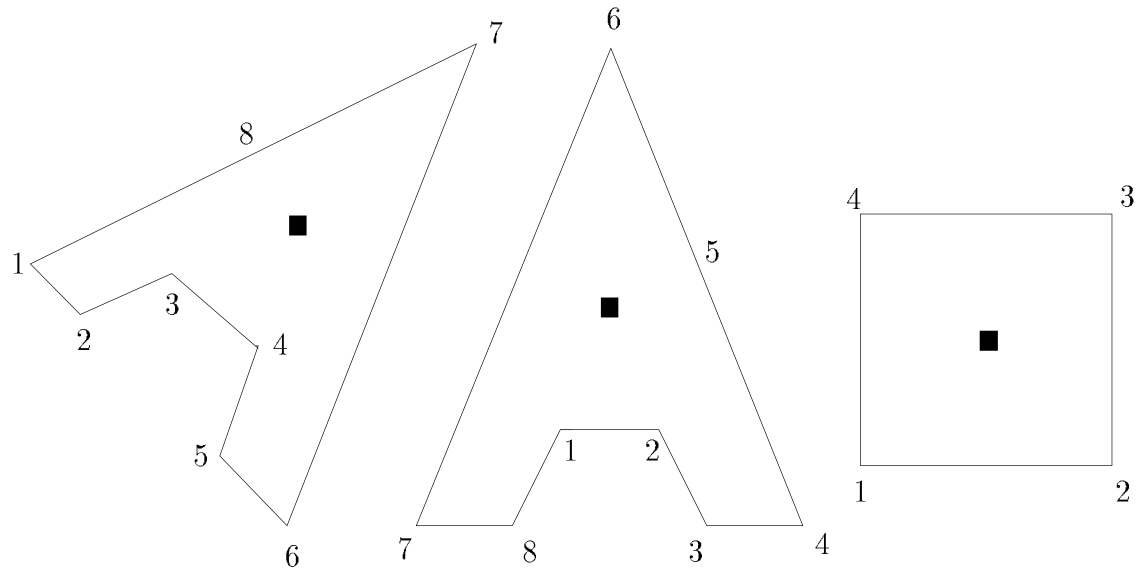
Problem A: Bumpy Objects



Object 1 Object 2 Square

Consider objects such as these. They are polygons, specified by the coordinates of a centre of mass and their vertices. In the figure, centres of mass are shown as black squares. The vertices will be numbered consecutively anti-clockwise as shown.

An object can be rotated to stand stably if two vertices can be found that can be joined by a straight line that does not intersect the object, and, when this line is horizontal, the centre of mass lies above the line and strictly between its endpoints. There are typically many stable positions and each is defined by one of these lines known as its base line. A base line, and its associated stable position, is identified by the highest numbered vertex touched by that line.

Write a program that will determine the stable position that has the lowest numbered base line. Thus for the above objects, the desired base lines would be 6 for object 1, 6 for object 2 and 2 for the square. You may assume that the objects are possible, that is they will be represented as non self-intersecting polygons, although they may well be concave.

Successive lines of a data set will contain: a string of less than 20 characters identifying the object; the coordinates of the centre of mass; and the coordinates of successive points terminated by two zeroes (0 0), on one or more lines as necessary. There may be successive data sets (objects). The end of data will be defined by the string '#'.

Output will consist of the identification string followed by the number of the relevant base line.

Sample input

```
Square
2 2
1 1  3 1  3 3  1 3 0 0
#
```

Sample output

```
1                   22
Object1              6
Object2              6
Square               2
```

Problem B: The Dole Queue

In a serious attempt to downsize (reduce) the dole queue, The New National Green Labour Rhinoceros Party has decided on the following strategy. Every day all dole applicants will be placed in a large circle, facing inwards. Someone is arbitrarily chosen as number 1, and the rest are numbered counter-clockwise up to N (who will be standing on 1's left). Starting from 1 and moving counter-clockwise, one labour official counts off k applicants, while another official starts from N and moves clockwise, counting m applicants. The two who are chosen are then sent off for retraining; if both officials pick the same person she (he) is sent off to become a politician. Each official then starts counting again at the next available person and the process continues until no-one is left. Note that the two victims (sorry, trainees) leave the ring simultaneously, so it is possible for one official to count a person already selected by the other official.

Write a program that will successively read in (in that order) the three numbers (N, k and m; k, m $>$ 0, 0 $<$ N $<$ 20) and determine the order in which the applicants are sent off for retraining. Each set of three numbers will be on a separate line and the end of data will be signalled by three zeroes (0 0 0).

For each triplet, output a single line of numbers specifying the order in which people are chosen. Each number should be in a field of 3 characters. For pairs of numbers list the person chosen by the counter-clockwise official first. Separate successive pairs (or singletons) by commas (but there should not be a trailing comma). Example:

Sample input

```
10 4 3
0 0 0
```

Sample output

```
△△4△△8,△△9△△5,△△3△△1,△△2△△6,△10,△△7
```

where △ represents a space.

Problem C: Loglan—A Logical Language

Loglan is a synthetic speakable language designed to test some of the fundamental problems of linguistics, such as the Sapir Whorf hypothesis. It is syntactically unambiguous, culturally neutral and metaphysically parsimonious. What follows is a gross over-simplification of an already very small grammar of some 200 rules.

Loglan sentences consist of a series of words and names, separated by spaces, and are terminated by a period (.). Loglan words all end with a vowel; names, which are derived extra-linguistically, end with a consonant. Loglan words are divided into two classes—little words which specify the structure of a sentence, and predicates which have the form CCVCV or CVCCV where C represents a consonant and V represents a vowel (see examples later).

The subset of Loglan that we are considering uses the following grammar:

A ⇒ a | e | i | o | u
MOD ⇒ ga | ge | gi | go | gu
BA ⇒ ba | be | bi | bo | bu
DA ⇒ da | de | di | do | du
LA ⇒ la | le | li | lo | lu
NAM ⇒ {all names}
PREDA ⇒ {all predicates}
<sentence> ⇒ <statement> | <predclaim>
<predclaim> ⇒ <predname> BA <preds> | DA <preds>
<preds> ⇒ <predstring> | <preds> A <predstring>
<predname> ⇒ LA <predstring> | NAM
<predstring> ⇒ PREDA | <predstring> PREDA
<statement> ⇒ <predname> <verbpred> <predname> | <predname> <verbpred>
<verbpred> ⇒ MOD <predstring>

Write a program that will read a succession of strings and determine whether or not they are correctly formed Loglan sentences.

Each Loglan sentence will start on a new line and will be terminated by a period (.). The sentence may occupy more than one line and words may be separated by more than one space. The input will be terminated by a line containing a single '#'. You can assume that all words will be correctly formed.

Output will consist of one line for each sentence containing either 'Good' or 'Bad!'.

Sample input

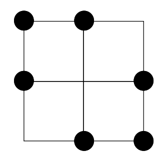
```
la mutce bunbo mrenu bi ditca.
la fumna bi le mrenu.
djan ga vedma le negro ketpi.
#
```

Sample output

```
Good
Bad!
Good
```

Problem D: No Rectangles

Consider a grid such as the one shown. We wish to mark k intersections in each of n rows and n columns in such a way that no 4 of the selected intersections form a rectangle with sides parallel to the grid. Thus for $k = 2$ and $n = 3$, a possible solution is:



It can easily be shown that for any given value of k, $k^2 - k + 1$ is a lower bound on the value of n, and it can be shown further that n need never be larger than this.

Write a program that will find a solution to this problem for $k = 12$, $n = 133$.

Output will consist of n lines of k points indicating the selected points on that line.

Example: if the problem had called for a solution to the problem for $k = 2$, $n = 3$; then the output could look like this:

Sample output

```
1 2
1 3
2 3
```

Problem E: Ugly Numbers

Ugly numbers are numbers whose only prime factors are 2, 3 or 5. The sequence

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...

shows the first 11 ugly numbers. By convention, 1 is included.

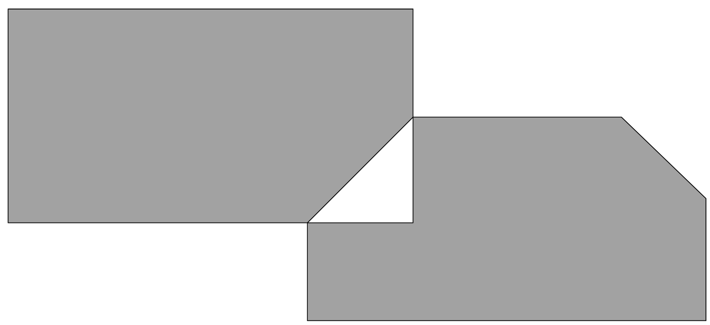
Write a program to find and print the 1500'th ugly number. There is no input to this program. Output should consist of a single line as shown below, with <number> replaced by the number computed.

Sample output

```
The 1500'th ugly number is <number>.
```

Problem F: Polygons

Given two convex polygons, they may or may not overlap. If they do overlap, they will do so to differing degrees and in different ways. Write a program that will read in the coordinates of the corners of two convex polygons and calculate the 'exclusive or' of the two areas, that is the area that is bounded by exactly one of the polygons. The desired area is shaded in the following diagram:



Input will consist of pairs of lines each containing the number of vertices of the polygon, followed by that many pairs of integers representing the x,y coordinates of the corners in a clockwise direction. All the coordinates will be positive integers less than 100. For each pair of polygons (pair of lines in the data file), your program should print out the desired area correct to two decimal places. The input will end with a line containing a zero (0).

Output will consist of a single line containing the desired area written as a succession of eight (8) digit fields with two (2) digits after the decimal point. There will not be enough cases to need more than one line.

Sample input

```
3  5 5  8 1  2 3
3  5 5  8 1  2 3
4  1 2  1 4  5 4  5 2
6  6 3  8 2  8 1  4 1  4 2  5 3
0
```

Sample output

```
△△△△0.00△△△13.50
```

where △ represents a single space.

Problem G: Street Numbers

A computer programmer lives in a street with houses numbered consecutively (from 1) down one side of the street. Every evening she walks her dog by leaving her house and randomly turning left or right and walking to the end of the street and back. One night she adds up the street numbers of the houses she passes (excluding her own). The next time she walks the other way she repeats this and finds, to her astonishment, that the two sums are the same. Although this is determined in part by her house number and in part by the number of houses in the street, she nevertheless feels that this is a desirable property for her house to have and decides that all her subsequent houses should exhibit it.

Write a program to find pairs of numbers that satisfy this condition. To start your list the first two pairs are: (house number, last number):

```
         6         8
        35        49
```

There is no input for this program. Output will consist of 10 lines each containing a pair of numbers, each printed left justified in a field of width 10 (as shown above).

Problem H: Telephone Tangles

A large company wishes to monitor the cost of phone calls made by its personnel. To achieve this the PABX logs, for each call, the number called (a string of up to 15 digits) and the duration in minutes. Write a program to process this data and produce a report specifying each call and its cost, based on standard Telecom charges.

International (IDD) numbers start with two zeroes (00) followed by a country code (1–3 digits) followed by a subscriber's number (4–10 digits). National (STD) calls start with one zero (0) followed by an area code (1–5 digits) followed by the subscriber's number (4–7 digits). The price of a call is determined by its destination and its duration. Local calls start with any digit other than 0 and are free.

Input will be in two parts. The first part will be a table of IDD and STD codes, localities and prices as follows:

Code△Locality name$price in cents per minute

where △ represents a space. Locality names are 25 characters or less. This section is terminated by a line containing 6 zeroes (000000).

The second part contains the log and will consist of a series of lines, one for each call, containing the number dialled and the duration. The file will be terminated a line containing a single #. The numbers will not necessarily be tabulated, although there will be at least one space between them. Telephone numbers will not be ambiguous.

Output will consist of the called number, the country or area called, the subscriber's number, the duration, the cost per minute and the total cost of the call, as shown below. Local calls are costed at zero. If the number has an invalid code, list the area as "Unknown" and the cost as −1.00.

Sample input

```
088925 Broadwood$81
03 Arrowtown$38
0061 Australia$140
000000
031526        22
0061853279  3
0889256287213   122
779760 1
002832769 5
#
```

Sample output

```
1              17               51   56    62     69
031526         Arrowtown      1526   22  0.38   8.36
0061853279     Australia    853279    3  1.40   4.20
0889256287213  Broadwood   6287213  122  0.81  98.82
779760         Local        779670    1  0.00   0.00
002832769      Unknown                5        -1.00
```