Problem 1—UND

Professor Plum likes it when MICS is hosted by the University of North Dakota since they hosted the first Symposium in 1967. He wants you to write a program to generate ASCII art printing "UND" vertically for a sign to hang on his door. Since he is unsure of the door's dimensions, he wants your program to take as input a positive integer scaling factor. The first several scaling factors with corresponding letter dimensions (height x width) are specified by the following table:

Scaling Factor	U and N Letter Dimension (# chars × # chars)	D Letter Dimension (# characters × # characters)	Line Width of Letters (# characters)	Blank Lines Between Letters U and N	Blank Lines Between Letters N and D
1	3 × 5	4 × 5	1	1	0
2	5 × 10	6 × 10	2	2	1
3	7 × 15	8 × 15	3	3	2
4	9 × 20	10 × 20	4	4	3
5	11 × 25	12 × 25	5	5	4

A scaling factor of 1 would produce:

```
|   |
|   |
\___/

|\  |
| \ |
|  \|

 ___
|   \
|   |
|___/
```

A scaling factor of 2 would produce:

```
||      ||
||      ||
||      ||
||______||
\\______//


||\\    ||
||  \\  ||
||   \\ ||
||    \\||
||     \\||


  ______
||______\\
||      ||
||      ||
||______||
||______//
```

Input Format

The input contains a single line with a positive integer scaling factor for the sign.

Output Format

The output should contain the ASCII art for the sign corresponding to the scaling factor specified by the input.

Input Sample

```
4
```

Output Sample

```
|||| ·········· ||||<EOLN>
|||| ·········· ||||<EOLN>
|||| ·········· ||||<EOLN>
|||| ·········· ||||<EOLN>
|||| ·········· ||||<EOLN>
||||_____________||||<EOLN>
||||_____________||||<EOLN>
||||_____________||||<EOLN>
\\\\_____________////<EOLN>
<EOLN>
<EOLN>
<EOLN>
<EOLN>
|||| \\\\ ····· ||||<EOLN>
|||| ·\\\\ ····· ||||<EOLN>
|||| ··\\\\ ····· ||||<EOLN>
|||| ···\\\\ ····· ||||<EOLN>
|||| ····\\\\ ····· ||||<EOLN>
|||| ·····\\\\ ····· ||||<EOLN>
|||| ·····\\\\ ····· ||||<EOLN>
|||| ·····\\\\ ····· ||||<EOLN>
|||| ·····\\\\ ····· ||||<EOLN>
|||| ·····\\\\||||<EOLN>
<EOLN>
<EOLN>
<EOLN>
    _____________<EOLN>
||||_____________\\\\<EOLN>
||||_____________||||<EOLN>
||||_____________||||<EOLN>
||||·····  ·····||||<EOLN>
||||·····  ·····||||<EOLN>
||||_____________||||<EOLN>
||||_____________||||<EOLN>
||||_____________||||<EOLN>
||||_____________////<EOLN>
<EOF>
```

← NOTICE THE DOTS (‘ ·’) REPRESENT BLANK SPACES
← AND <EOLN> REPRESENTS END-OF-LINE.
← THERE SHOULD BE NO DOTS AND “<EOLN>” STRINGS IN
← YOUR ACTUAL OUTPUT

Problem 2—Stamp Out Holes

Professor Plum is somewhat compulsive about postage stamps when he travels because he likes to mail home souvenirs from the road. Before leaving for MICS, he packed his postage scale and filled his pocket with a hand full of stamps from home. As he drives the van, he wants you to inventory his stamps and write a program to determine the least amount of postage that can't be made using his collection of stamps, and how many amounts between 0 and the total of all his stamps can't be made from the pocketful of stamps.

Input Format

The input consists of two lines. The first line contains the current value of a "Forever" stamp. The second line lists one or more positive integer stamp values or the letter F (for "Forever" stamps). There are at most 25 stamp value(s) on the second line which are all separated by one space.

Output Format

For the given input, compute the least nonnegative integer amount that cannot be made exactly using stamp values from the input list. Note that it must be greater than zero (which can always be made using no stamps) and not greater than the total of the stamps plus one. Also compute the number of amounts between zero and the total of the stamps that cannot be made exactly using stamp values from the input list. Format the output as shown in the output sample below.

Input Sample

```
49
20 F 1 4 34 3 5 1 F 20
```

Output Sample

```
The least amount that cannot be made exactly is 15
The number of amounts between 0 and 186 that cannot be made exactly is 10
```

Problem 3—Steganography

Professor Plum thinks of himself as an amateur spy because he likes to dabble in steganography. *Steganography* is the process of concealing a secret message, image, or video within another message, image, or video. His latest scheme is to send a text file of non-negative integers one per line. The integers are nearly random, except each contains an embedded character code corresponding to a character in the secret message. If each integer is divided by 30, then the remainder is a numeric character code corresponding to characters according to:

- the character codes 1, 2, 3, 4, 5 …, 21, 22, 23, 24, 25, 26 correspond to the letters `'A'`, `'B'`, `'C'`, `'D'`, `'E'`, …, `'U'`, `'V'`, `'W'`, `'X'`, `'Y'`, `'Z'` respectively,
- the character code 0 corresponds to `' '` (space) character,
- the character code 27 corresponds to `'.'` (period) character,
- the character code 28 corresponds `'?'` (question mark) character, and
- the character code 29 corresponds to the end-of-line.

Input Format

The input consists of multiple lines with each containing a single non-negative integer corresponding to a character in the secret message as described above.

Output Format

The sequence of decoded characters including end-of-lines.

Input Sample

```
31
0
33
61
62
117
30
93
121
14
29
39
50
30
2
65
88
59
```

Output Sample

```
A CAB. CAN
IT BE?
```

Problem 4—Crazy Eight Divisors

Professor Plum's favorite number is eight. The eight divisors of 24 are 1, 2, 3, 4, 6, 8, 12 and 24.

The ten numbers not exceeding 100 having exactly eight divisors are 24, 30, 40, 42, 54, 56, 66, 70, 78 and 88.

Let f(n) be the count of numbers not exceeding n with exactly eight divisors.

Professor Plum has hand-calculated f(100) = 10 and f(1000) = 180, but wants you to write a program for bigger values of n up to 2,000,000.

Input Format

The input consists of a single line containing a non-negative integer that is $\leq$ 2,000,000.

Output Format

The output consists of a single line formatted as shown below for the input n = 1000 and f(1000) = 180. Notice the period at the end of the output line.

Input Sample

```
1000
```

Output Sample

```
The count of numbers not exceeding 1000 with exactly eight divisors is 180.
```

Problem 5—Valley Sort

Professor Plum likes to bicycle in the Rocky Mountains during his summer vacation. He typically gets dropped off at the top of a mountain and bikes to the valley below. While writing an array question for his final examination in CS 101, he invents the notion of a *valley sort* where the first half of the array is in descending order and last half of the array is in ascending order. More specifically, the largest item is in the first index, the second largest item is in the last index, the third largest item is in the second index, the fourth largest item is in the next to last index, etc.

For example, an array initially order as: 20, 45, 30, 5, 15, 50, 10, 35 would be valley sorted to:

50, 35, 20, 10, 5, 15, 30, 45.

Input Format

The first line of the input file contains an integer count of the number of items to valley sort. The remaining lines will contain one integer per line.

Output Format

The first line of the output file should contain an integer count of the number of items valley sorted. The remaining lines will contain one integer per line in valley-sorted order.

Input Sample

```
8
20
45
30
5
15
50
10
35
```

Output Sample

```
8
50
35
20
10
5
15
30
45
```

Problem 6—Expanding Password

Professor Plum has a hard time remembering passwords, so he has used the same password `9P8L7U6M@52` forever. The IT department has warned him that they are going to require much longer passwords on the Monday after MICS, but they have not told him the minimum length yet. Professor Plum wants to be prepared and possibly help others in the same situation. He wants you to write a program to lengthen any password containing decimal digits (0 to 9) by converting each decimal digit to a lower base. For example his password `9P8L7U6M@52` with each decimal digit converted to base-2 (i.e., binary) would give the longest password of `1001P1000L111U110M@10110`.

Since 24-digits might be longer than the IT department's new password length, he wants to use the **largest** base necessary to meet the new IT specified password length. If the password cannot be lengthened enough, then the base-2 conversion will be used with a suffix of enough '`@`' characters to reach the desired length. For example his password `9P8L7U6M@52` expanded to length 30 would be `1001P1000L111U110M@10110@@@@@@`.

Input Format

The input consists of multiple lines with each containing a password-length pair separated by a space.

Output Format

For each password-length pair in the input, a single line of output should be produced. Each output line should be formatted as shown below.

Input Sample

```
9P8L7U6M@52 8
9P8L7U6M@52 13
9P8L7U6M@52 20
9P8L7U6M@52 30
9P8L7U6M@52 16
=MiCs2015 10
Cat 5
```

Output Sample

```
Old password 9P8L7U6M@52 expanded to length 8 is 9P8L7U6M@52
Old password 9P8L7U6M@52 expanded to length 13 is 11P10L7U6M@52
Old password 9P8L7U6M@52 expanded to length 20 is 1001P1000L111U110M@10110
Old password 9P8L7U6M@52 expanded to length 30 is 1001P1000L111U110M@10110@@@@@@
Old password 9P8L7U6M@52 expanded to length 16 is 14P13L12U11M@102
Old password =MiCs2015 expanded to length 10 is =MiCs20110
Old password Cat expanded to length 5 is Cat@@
```

Problem 7—B/W Photo Copyright

Professor Plum takes a lot of panoramic black-and-white photographs and posts them on his website. He is concerned about other people posting copies of his photographs without attribution to him. He is especially concerned about people posting cropped portions of his images, so he wants a program to detect if a candidate photograph C could be generated by cropping his own photograph T. If the photograph C is found within photograph T, then the program should report the row and column indexes within T where the upper-left corner of C was found. Assume the row and column indexes start at 0.

Input Format

The input consists of two black-and-white photographs: C followed by T. The first line of the file will consist of two positive integers: h_C and w_C which are the height and width of photograph C. h_C lines follow with each line containing w_C integer values separated by single spaces. These integer values all range from 0 to 255 and represent the intensity value of pixels within the black-and-white photograph.

After the last line of photograph C, is the dimension line for photograph T containing two positive integers: h_T and w_T which are the height and width of photograph T. h_T lines follow with each line containing w_T integer values separated by single spaces.

Output Format

Two outcomes are possible with each producing a single line. If C is not found in T, then the output line should be:

```
Photograph C was not found in T
```

If C is found in T starting at say row 25 and column 10, then the output line should be:

```
Photograph C was found in T starting at row 25 and column 10
```

Note: Input Sample and Output Sample are on the next page

Input Sample

```
10 15
11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
20 25
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 88 88 88
88 88 88 88 88 88 88 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 88 88 88
88 88 88 88 88 88 88 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 88 88 88
88 88 88 88 88 88 88 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 88 88 88
88 88 88 88 88 88 88 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 88 88 88
88 88 88 88 88 88 88 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 88 88 88
88 88 88 88 88 88 88 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 88 88 88
88 88 88 88 88 88 88 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 88 88 88
88 88 88 88 88 88 88 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 88 88 88
88 88 88 88 88 88 88 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88 88
```

Output Sample

```
Photograph C was found in T starting at row 2 and column 7
```