USACO Training Problems

---

http://train.usaco.org

Collected by Li

Contents

Chapter 1 Getting Started

Section 1.1

1.1.1 Your Ride is Here

It is a well-known fact that behind every good comet is a UFO. These UFOs often come to collect loyal supporters from here on Earth. Unfortunately, they only have room to pick up one group of followers on each trip. They do, however, let the groups know ahead of time which will be picked up for each comet by a clever scheme: they pick a name for the comet which, along with the name of the group, can be used to determine if it is a particular group's turn to go (who do you think names the comets?). The details of the matching scheme are given below; your job is to write a program which takes the names of a group and a comet and then determines whether the group should go with the UFO behind that comet.

Both the name of the group and the name of the comet are converted into a number in the following manner: the final number is just the product of all the letters in the name, where "A" is 1 and "Z" is 26. For instance, the group "USACO" would be 21 * 19 * 1 * 3 * 15 = 17955. If the group's number mod 47 is the same as the comet's number mod 47, then you need to tell the group to get ready! (Remember that "a mod b" is the remainder left over after dividing a by b; 34 mod 10 is 4.)

Write a program which reads in the name of the comet and the name of the group and figures out whether according to the above scheme the names are a match, printing "GO" if they match and "STAY" if not. The names of the groups and the comets will be a string of capital letters with no spaces or punctuation, up to 6 characters long.

Examples:

```
Input   Output
COMETQ  GO
HVNGAT
ABSTAR  STAY
USACO
```

PROGRAM NAME: ride

This means that you fill in your header with:

```
PROG: ride
```

INPUT FORMAT

Line 1: An upper case character string of length 1..6 that is the name of the comet.

Line 2: An upper case character string of length 1..6 that is the name of the group.

NOTE: The input file has a newline at the end of each line but does not have a "return". Sometimes, programmers code for the Windows paradigm of "return" followed by "newline"; don't do that! Use simple input routines like "readln" (for Pascal) and, for C/C++, "fscanf" and "fid>>string".

SAMPLE INPUT (file ride.in)

```
COMETQ
HVNGAT
```

OUTPUT FORMAT

A single line containing either the word "GO" or the word "STAY".

SAMPLE OUTPUT (file ride.out)

```
GO
```

1.1.2 Greedy Gift Givers

A group of NP ($2 \le NP \le 10$) uniquely named friends has decided to exchange gifts of money. Each of these friends might or might not give some money to any or all of the other friends. Likewise, each friend might or might not receive money from any or all of the other friends. Your goal in this problem is to deduce how much more money each person gives than they receive.

The rules for gift-giving are potentially different than you might expect. Each person sets aside a certain amount of money to give and divides this money evenly among all those to whom he or she is giving a gift. No fractional money is available, so dividing 3 among 2 friends would be 1 each for the friends with 1 left over -- that 1 left over stays in the giver's "account".

In any group of friends, some people are more giving than others (or at least may have more acquaintances) and some people have more money than others.

Given a group of friends, no one of whom has a name longer than 14 characters, the money each person in the group spends on gifts, and a (sub)list of friends to whom each person gives gifts, determine how much more (or less) each person in the group gives than they receive.

IMPORTANT NOTE

The grader machine is a Linux machine that uses standard Unix conventions: end of line is a single character often known as '\n'. This differs from Windows, which ends lines with two charcters, '\n' and '\r'. Do not let your program get trapped by this!

PROGRAM NAME: gift1

INPUT FORMAT

Line 1: The single integer, NP

Lines 2..NP+1: Each line contains the name of a group member

Lines NP+2..end: NP groups of lines organized like this:

The first line in the group tells the person's name who will be giving gifts.

The second line in the group contains two numbers: The initial amount of money (in the range 0..2000) to be divided up into gifts by the giver and then the number of people to whom the giver will give gifts, NG_i ($0 \le NG_i \le NP-1$).

If NG_i is nonzero, each of the next NG_i lines lists the the name of a recipient of a gift.

SAMPLE INPUT (file gift1.in)

```
5
dave
laura
owen
vick
amr
dave
200 3
laura
owen
vick
owen
500 1
dave
amr
150 2
vick
owen
laura
0 2
```

```
amr
vick
vick
0 0
```

OUTPUT FORMAT

The output is NP lines, each with the name of a person followed by a single blank followed by the net gain or loss (final_money_value - initial_money_value) for that person. The names should be printed in the same order they appear on line 2 of the input.

All gifts are integers. Each person gives the same integer amount of money to each friend to whom any money is given, and gives as much as possible that meets this constraint. Any money not given is kept by the giver.

SAMPLE OUTPUT (file gift1.out)

```
dave 302
laura 66
owen -359
vick 141
amr -150
```

1.1.3 Friday the Thirteenth

Is Friday the 13th really an unusual event?

That is, does the 13th of the month land on a Friday less often than on any other day of the week? To answer this question, write a program that will compute the frequency that the 13th of each month lands on Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday over a given period of N years. The time period to test will be from January 1, 1900 to December 31, 1900+N-1 for a given number of years, N. N is non-negative and will not exceed 400.

There are few facts you need to know before you can solve this problem:

- January 1, 1900 was on a Monday.
- Thirty days has September, April, June, and November, all the rest have 31 except for February which has 28 except in leap years when it has 29.
- Every year evenly divisible by 4 is a leap year (1992 = 4*498 so 1992 will be a leap year, but the year 1990 is not a leap year)
- The rule above does not hold for century years. Century years divisible by 400 are leap years, all other are not. Thus, the century years 1700, 1800, 1900 and 2100 are not leap years, but 2000 is a leap year.

Do not use any built-in date functions in your computer language.

Don't just precompute the answers, either, please.

PROGRAM NAME: friday

INPUT FORMAT

One line with the integer N.

SAMPLE INPUT (file friday.in)

```
20
```

OUTPUT FORMAT

Seven space separated integers on one line. These integers represent the number of times the 13th falls on Saturday, Sunday, Monday, Tuesday, ..., Friday.

SAMPLE OUTPUT (file friday.out)

```
36 33 34 33 35 35 34
```

1.1.4 Broken Necklace

You have a necklace of N red, white, or blue beads (3<=N<=350) some of which are red, others blue, and others white, arranged at random. Here are two examples for n=29:

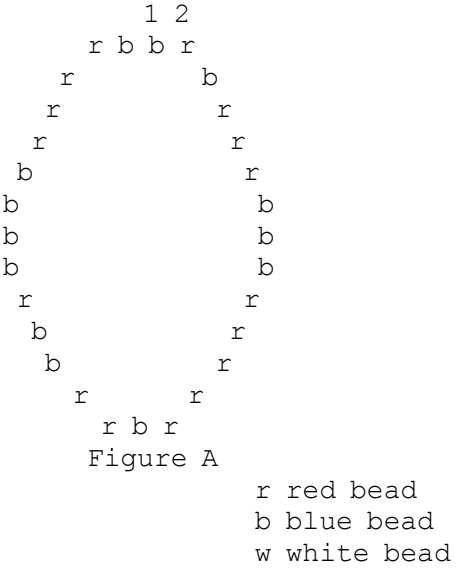


Figure A

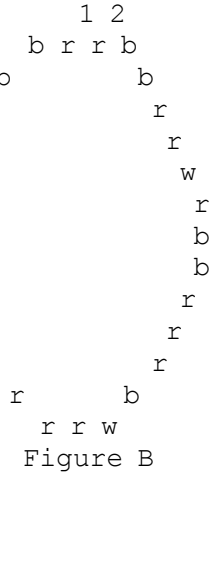


Figure B

r red bead
b blue bead
w white bead

The beads considered first and second in the text that follows have been marked in the picture.

The configuration in Figure A may be represented as a string of b's and r's, where b represents a blue bead and r represents a red one, as follows: brbrrrbbbrrrrrbrrbbrbbbbrrrrb .

Suppose you are to break the necklace at some point, lay it out straight, and then collect beads of the same color from one end until you reach a bead of a different color, and do the same for the other end (which might not be of the same color as the beads collected before this).

Determine the point where the necklace should be broken so that the most number of beads can be collected.

Example

For example, for the necklace in Figure A, 8 beads can be collected, with the breaking point either between bead 9 and bead 10 or else between bead 24 and bead 25.

In some necklaces, white beads had been included as shown in Figure B above. When collecting beads, **a white bead that is encountered may be treated as either red or blue and then painted with the desired color**. The string that represents this configuration will include the three symbols r, b and w.

Write a program to determine the largest number of beads that can be collected from a supplied necklace.

PROGRAM NAME: beads

INPUT FORMAT

Line 1: N, the number of beads

Line 2: a string of N characters, each of which is r, b, or w

SAMPLE INPUT (file beads.in)

```
29
wwwbbrwrbrbrrbrbrwrwwrbwrwrrb
```

OUTPUT FORMAT

A single line containing the maximum of number of beads that can be collected from the supplied necklace.

SAMPLE OUTPUT (file beads.out)

```
11
```

OUTPUT EXPLANATION

Consider two copies of the beads (kind of like being able to runaround the ends). The string of 11 is marked.

```
wwwbbrwrbrbrrbrbrwrwwrbwrwrrb wwwbbrwrbrbrrbrbrwrwwrbwrwrrb
                         ****** *****
                         rrrrrb bbbbb  <-- assignments
                          5 x r  6 x b  <-- 11 total
```

Section 1.2

1.2.1 Milking Cows

Three farmers rise at 5 am each morning and head for the barn to milk three cows. The first farmer begins milking his cow at time 300 (measured in seconds after 5 am) and ends at time 1000. The second farmer begins at time 700 and ends at time 1200. The third farmer begins at time 1500 and ends at time 2100. The longest continuous time during which at least one farmer was milking a cow was 900 seconds (from 300 to 1200). The longest time no milking was done, between the beginning and the ending of all milking, was 300 seconds (1500 minus 1200).

Your job is to write a program that will examine a list of beginning and ending times for N (1 <= N <= 5000) farmers milking N cows and compute (in seconds):

- The longest time interval at least one cow was milked.
- The longest time interval (after milking starts) during which no cows were being milked.

PROGRAM NAME: milk2

INPUT FORMAT

Line 1: The single integer

Lines 2..N+1: Two non-negative integers less than 1000000, the starting and ending time in seconds after 0500

SAMPLE INPUT (file milk2.in)

```
3
300 1000
700 1200
1500 2100
```

OUTPUT FORMAT

A single line with two integers that represent the longest continuous time of milking and the longest idle time.

SAMPLE OUTPUT (file milk2.out)

```
900 300
```

1.2.2 Transformations

A square pattern of size N x N (1 <= N <= 10) black and white square tiles is transformed into another square pattern. Write a program that will recognize the minimum transformation that has been applied to the original pattern given the following list of possible transformations:

- #1: 90 Degree Rotation: The pattern was rotated clockwise 90 degrees.
- #2: 180 Degree Rotation: The pattern was rotated clockwise 180 degrees.
- #3: 270 Degree Rotation: The pattern was rotated clockwise 270 degrees.
- #4: Reflection: The pattern was reflected horizontally (turned into a mirror image of itself by reflecting around a vertical line in the middle of the image).
- #5: Combination: The pattern was reflected horizontally and then subjected to one of the rotations (#1-#3).
- #6: No Change: The original pattern was not changed.
- #7: Invalid Transformation: The new pattern was not obtained by any of the above methods.

In the case that more than one transform could have been used, choose the one with the minimum number above.

PROGRAM NAME: transform

INPUT FORMAT

Line 1: A single integer, N

Line 2..N+1: N lines of N characters (each either `@' or `-'); this is the square before transformation

Line N+2..2*N+1: N lines of N characters (each either `@' or `-'); this is the square after transformation

SAMPLE INPUT (file transform.in)

```
3
@-@
---
@@-
@-@
@--
--@
```

OUTPUT FORMAT

A single line containing the the number from 1 through 7 (described above) that categorizes the transformation required to change from the `before' representation to the `after' representation.

SAMPLE OUTPUT (file transform.out)

```
1
```

1.2.3 Name That Number

Among the large Wisconsin cattle ranchers, it is customary to brand cows with serial numbers to please the Accounting Department. The cow hands don't appreciate the advantage of this filing system, though, and wish to call the members of their herd by a pleasing name rather than saying, "C'mon, #4734, get along."

Help the poor cowhands out by writing a program that will translate the brand serial number of a cow into possible names uniquely associated with that serial number. Since the cow hands all have cellular saddle phones these days, use the standard Touch-Tone(R) telephone keypad mapping to get from numbers to letters (except for "Q" and "Z"):

```
2: A,B,C     5: J,K,L    8: T,U,V
3: D,E,F     6: M,N,O    9: W,X,Y
4: G,H,I     7: P,R,S
```

Acceptable names for cattle are provided to you in a file named "dict.txt", which contains a list of fewer than 5,000 acceptable cattle names (all letters capitalized). Take a cow's brand number and report which of all the possible words to which that number maps are in the given dictionary[i] which is supplied as dict.txt in the grading environment (and is sorted into ascending order).

For instance, the brand number 4734 produces all the following names:

```
GPDG GPDH GPDI GPEG GPEH GPEI GPFG GPFH GPFI GRDG GRDH GRDI
GREG GREH GREI GRFG GRFH GRFI GSDG GSDH GSDI GSEG GSEH GSEI
GSFG GSFH GSFI HPDG HPDH HPDI HPEG HPEH HPEI HPFG HPFH HPFI
HRDG HRDH HRDI HREG HREH HREI HRFG HRFH HRFI HSDG HSDH HSDI
HSEG HSEH HSEI HSFG HSFH HSFI IPDG IPDH IPDI IPEG IPEH IPEI
IPFG IPFH IPFI IRDG IRDH IRDI IREG IREH IREI IRFG IRFH IRFI
ISDG ISDH ISDI ISEG ISEH ISEI ISFG ISFH ISFI
```

As it happens, the only one of these 81 names that is in the list of valid names is "GREG".

Write a program that is given the brand number of a cow and prints all the valid names that can be generated from that brand number or ``NONE" if there are no valid names. Serial numbers can be as many as a dozen digits long.

[i] See http://ace.delos.com/usaco/namenumdict.txt.

PROGRAM NAME: namenum

INPUT FORMAT

A single line with a number from 1 through 12 digits in length.

SAMPLE INPUT (file namenum.in)

```
4734
```

OUTPUT FORMAT

A list of valid names that can be generated from the input, one per line, in ascending alphabetical order.

SAMPLE OUTPUT (file namenum.out)

```
GREG
```

1.2.4 Palindromic Squares

Rob Kolstad

Palindromes are numbers that read the same forwards as backwards. The number 12321 is a typical palindrome.

Given a number base B (2 <= B <= 20 base 10), print all the integers N (1 <= N <= 300 base 10) such that the square of N is palindromic when expressed in base B; also print the value of that palindromic square. Use the letters 'A', 'B', and so on to represent the digits 10, 11, and so on.

Print both the number and its square in base B.

PROGRAM NAME: palsquare

INPUT FORMAT

A single line with B, the base (specified in base 10).

SAMPLE INPUT (file palsquare.in)

```
10
```

OUTPUT FORMAT

Lines with two integers represented in base B. The first integer is the number whose square is palindromic; the second integer is the square itself.

SAMPLE OUTPUT (file palsquare.out)

```
1 1
2 4
3 9
11 121
22 484
26 676
101 10201
111 12321
121 14641
202 40804
212 44944
264 69696
```

1.2.5 Dual Palindromes

Mario Cruz (Colombia) & Hugo Rickeboer (Argentina)

A number that reads the same from right to left as when read from left to right is called a palindrome. The number 12321 is a palindrome; the number 77778 is not. Of course, palindromes have neither leading nor trailing zeroes, so 0220 is not a palindrome.

The number 21 (base 10) is not palindrome in base 10, but the number 21 (base 10) is, in fact, a palindrome in base 2

(10101).

Write a program that reads two numbers (expressed in base 10):

- N (1 <= N <= 15)
- S (0 < S < 10000)

and then finds and prints (in base 10) the first N numbers strictly greater than S that are palindromic when written in two or more number bases (2 <= base <= 10).

Solutions to this problem do not require manipulating integers larger than the standard 32 bits.

PROGRAM NAME: dualpal

INPUT FORMAT

A single line with space separated integers N and S.

SAMPLE INPUT (file dualpal.in)

```
3 25
```

OUTPUT FORMAT

N lines, each with a base 10 number that is palindromic when expressed in at least two of the bases 2..10. The numbers should be listed in order from smallest to largest.

SAMPLE OUTPUT (file dualpal.out)

```
26
27
28
```

Section 1.3

1.3.1 Mixing Milk

Since milk packaging is such a low margin business, it is important to keep the price of the raw product (milk) as low as possible. Help Merry Milk Makers get the milk they need in the cheapest possible manner.

The Merry Milk Makers company has several farmers from which they may buy milk, and each one has a (potentially) different price at which they sell to the milk packing plant. Moreover, as a cow can only produce so much milk a day, the farmers only have so much milk to sell per day. Each day, Merry Milk Makers can purchase an integral amount of milk from each farmer, less than or equal to the farmer's limit.

Given the Merry Milk Makers' daily requirement of milk, along with the cost per gallon and amount of available milk for each farmer, calculate the minimum amount of money that it takes to fulfill the Merry Milk Makers' requirements.

Note: The total milk produced per day by the farmers will be sufficient to meet the demands of the Merry Milk Makers.

PROGRAM NAME: milk

INPUT FORMAT

Line 1:	Two integers, N and M. The first value, N, (0 <= N <= 2,000,000) is the amount of milk that Merry Milk Makers wants per day. The second, M, (0 <= M <= 5,000) is the number of farmers that they may buy from.
Lines 2 through M+1:	The next M lines each contain two integers, P_i and A_i. P_i (0 <= P_i <= 1,000) is price in cents that farmer i charges. A_i (0 <= Ai <= 2,000,000) is the amount of milk that farmer i can sell to Merry Milk Makers per day.

SAMPLE INPUT (file milk.in)

```
100 5
5 20
9 40
```

```
3 10
8 80
6 30
```

OUTPUT FORMAT

A single line with a single integer that is the minimum price that Merry Milk Makers can get their milk at for one day.

SAMPLE OUTPUT (file milk.out)

```
630
```

1.3.2 Barn Repair

It was a dark and stormy night that ripped the roof and gates off the stalls that hold Farmer John's cows. Happily, many of the cows were on vacation, so the barn was not completely full.

The cows spend the night in stalls that are arranged adjacent to each other in a long line. Some stalls have cows in them; some do not. All stalls are the same width.

Farmer John must quickly erect new boards in front of the stalls, since the doors were lost. His new lumber supplier will supply him boards of any length he wishes, but the supplier can only deliver a small number of total boards. Farmer John wishes to minimize the total length of the boards he must purchase.

Given M (1 <= M <= 50), the maximum number of boards that can be purchased; S (1 <= S <= 200), the total number of stalls; C (1 <= C <= S) the number of cows in the stalls, and the C occupied stall numbers (1 <= stall_number <= S), calculate the minimum number of stalls that must be blocked in order to block all the stalls that have cows in them.

Print your answer as the total number of stalls blocked.

PROGRAM NAME: barn1

INPUT FORMAT

Line 1: M, S, and C (space separated)

Lines 2-C+1: Each line contains one integer, the number of an occupied stall.

SAMPLE INPUT (file barn1.in)

```
4 50 18
3
4
6
8
14
15
16
17
21
25
26
27
30
31
40
41
42
43
```

OUTPUT FORMAT

A single line with one integer that represents the total number of stalls blocked.

SAMPLE OUTPUT (file barn1.out)

```
25
```

[One minimum arrangement is one board covering stalls 3-8, one covering 14-21, one covering 25-31, and one covering

1.3.3 Calf Flac

It is said that if you give an infinite number of cows an infinite number of heavy-duty laptops (with very large keys), that they will ultimately produce all the world's great palindromes. Your job will be to detect these bovine beauties.

Ignore punctuation, whitespace, numbers, and case when testing for palindromes, but keep these extra characters around so that you can print them out as the answer; just consider the letters `A-Z' and `a-z'.

Find the largest palindrome in a string no more than 20,000 characters long. The largest palindrome is guaranteed to be at most 2,000 characters long before whitespace and punctuation are removed.

PROGRAM NAME: calfflac

INPUT FORMAT

A file with no more than 20,000 characters. The file has one or more lines which, when taken together, represent one long string. No line is longer than 80 characters (not counting the newline at the end).

SAMPLE INPUT (file calfflac.in)

```
Confucius say: Madam, I'm Adam.
```

OUTPUT FORMAT

The first line of the output should be the length of the longest palindrome found. The next line or lines should be the actual text of the palindrome (without any surrounding white space or punctuation but with all other characters) printed on a line (or more than one line if newlines are included in the palindromic text). If there are multiple palindromes of longest length, output the one that appears first.

SAMPLE OUTPUT (file calfflac.out)

```
11
Madam, I'm Adam
```

1.3.4 Prime Cryptarithm

The following cryptarithm is a multiplication problem that can be solved by substituting digits from a specified set of N digits into the positions marked with *. If the set of prime digits {2,3,5,7} is selected, the cryptarithm is called a PRIME CRYPTARITHM.

```
     * * *
  x    * *
   -------
     * * *        <-- partial product 1
    * * *         <-- partial product 2
    -------
    * * * *
```

Digits can appear only in places marked by `*'. Of course, leading zeroes are not allowed.

Note that the 'partial products' are as taught in USA schools. The first partial product is the product of the final digit of the second number and the top number. The second partial product is the product of the first digit of the second number and the top number.

Write a program that will find all solutions to the cryptarithm above for any subset of digits from the set {1,2,3,4,5,6,7,8,9}.

PROGRAM NAME: crypt1

INPUT FORMAT

Line 1: N, the number of digits that will be used

Line 2: N space separated digits with which to solve the cryptarithm

SAMPLE INPUT (file crypt1.in)

```
5
2 3 4 6 8
```

OUTPUT FORMAT

A single line with the total number of unique solutions. Here is the single solution for the sample input:

```
      2 2 2
    x   2 2
     ------
      4 4 4
    4 4 4
  ---------
    4 8 8 4
```

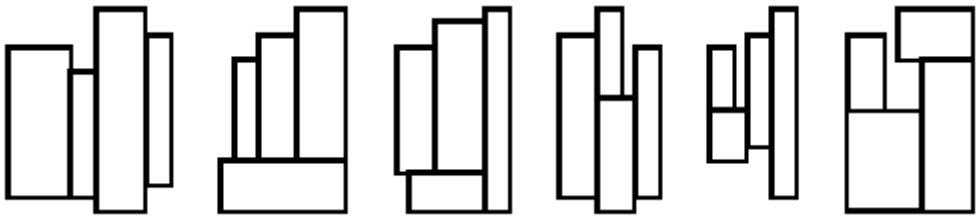
SAMPLE OUTPUT (file crypt1.out)

```
1
```

Section 1.4

1.4.1 Packing Rectangles

IOI 95



The six basic layouts of four rectangles

Four rectangles are given. Find the smallest enclosing (new) rectangle into which these four may be fitted without overlapping. By smallest rectangle, we mean the one with the smallest area.

All four rectangles should have their sides parallel to the corresponding sides of the enclosing rectangle. Figure 1 shows six ways to fit four rectangles together. These six are the only possible basic layouts, since any other layout can be obtained from a basic layout by rotation or reflection. Rectangles may be rotated 90 degrees during packing.

There may exist several different enclosing rectangles fulfilling the requirements, all with the same area. You must produce all such enclosing rectangles.

PROGRAM NAME: packrec

INPUT FORMAT

Four lines, each containing two positive space-separated integers that represent the lengths of a rectangle's two sides. Each side of a rectangle is at least 1 and at most 50.

SAMPLE INPUT (file packrec.in)

```
1 2
2 3
3 4
4 5
```

OUTPUT FORMAT

The output file contains one line more than the number of solutions. The first line contains a single integer: the minimum area of the enclosing rectangles. Each of the following lines contains one solution described by two numbers p and q with p<=q. These lines must be sorted in ascending order of p, and must all be different.

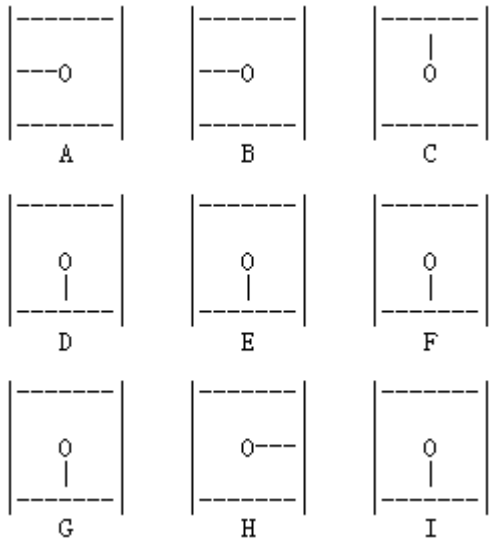
SAMPLE OUTPUT (file packrec.out)

```
40
4 10
5 8
```

1.4.2 The Clocks

IOI'94 - Day 2

Consider nine clocks arranged in a 3x3 array thusly:



The goal is to find a minimal sequence of moves to return all the dials to 12 o'clock. Nine different ways to turn the dials on the clocks are supplied via a table below; each way is called a move. Select for each move a number 1 through 9 which will cause the dials of the affected clocks (see next table) to be turned 90 degrees clockwise.

Move	Affected clocks
1	ABDE
2	ABC
3	BCEF
4	ADG
5	BDEFH
6	CFI
7	DEGH
8	GHI
9	EFHI

Example

Each number represents a time accoring to following table:

```
9 9 12       9 12 12       9 12 12       12 12 12      12 12 12
6 6 6  5 ->  9  9  9  8->  9  9  9  4 ->  12  9  9  9-> 12 12 12
6 3 6        6  6  6       9  9  9       12  9  9      12 12 12
```

[But this might or might not be the `correct' answer; see below.]

PROGRAM NAME: clocks

INPUT FORMAT

Lines 1-3: Three lines of three space-separated numbers; each number represents the start time of one clock, 3, 6, 9, or 12. The ordering of the numbers corresponds to the first example above.

SAMPLE INPUT (file clocks.in)

```
9 9 12
```

```
6 6 6
6 3 6
```

OUTPUT FORMAT

A single line that contains a space separated list of the shortest sequence of moves (designated by numbers) which returns all the clocks to 12:00. If there is more than one solution, print the one which gives the lowest number when the moves are concatenated (e.g., 5 2 4 6 < 9 3 1 1).

SAMPLE OUTPUT (file clocks.out)

```
4 5 8 9
```

1.4.3 Arithmetic Progressions

An arithmetic progression is a sequence of the form a, a+b, a+2b, ..., a+nb where n=0,1,2,3,... . For this problem, a is a non-negative integer and b is a positive integer.

Write a program that finds all arithmetic progressions of length n in the set S of bisquares. The set of bisquares is defined as the set of all integers of the form $p^2 + q^2$ (where p and q are non-negative integers).

TIME LIMIT: 5 secs

PROGRAM NAME: ariprog

INPUT FORMAT

Line 1: N (3 <= N <= 25), the length of progressions for which to search

Line 2: M (1 <= M <= 250), an upper bound to limit the search to the bisquares with 0 <= p,q <= M.

SAMPLE INPUT (file ariprog.in)

```
5
7
```

OUTPUT FORMAT

If no sequence is found, a singe line reading `NONE'. Otherwise, output one or more lines, each with two integers: the first element in a found sequence and the difference between consecutive elements in the same sequence. The lines should be ordered with smallest-difference sequences first and smallest starting number within those sequences first.

There will be no more than 10,000 sequences.

SAMPLE OUTPUT (file ariprog.out)

```
1 4
37 4
2 8
29 8
1 12
5 12
13 12
17 12
5 20
2 24
```

1.4.4 Mother's Milk

Farmer John has three milking buckets of capacity A, B, and C liters. Each of the numbers A, B, and C is an integer from 1 through 20, inclusive. Initially, buckets A and B are empty while bucket C is full of milk. Sometimes, FJ pours milk from one bucket to another until the second bucket is filled or the first bucket is empty. Once begun, a pour must be completed, of course. Being thrifty, no milk may be tossed out.

Write a program to help FJ determine what amounts of milk he can leave in bucket C when he begins with three buckets as

above, pours milk among the buckets for a while, and then notes that bucket A is empty.

PROGRAM NAME: milk3

INPUT FORMAT

A single line with the three integers A, B, and C.

SAMPLE INPUT (file milk3.in)

```
8 9 10
```

OUTPUT FORMAT

A single line with a sorted list of all the possible amounts of milk that can be in bucket C when bucket A is empty.

SAMPLE OUTPUT (file milk3.out)

```
1 2 8 9 10
```

SAMPLE INPUT (file milk3.in)

```
2 5 10
```

SAMPLE OUTPUT (file milk3.out)

```
5 6 7 8 9 10
```

Section 1.5

1.5.1 Number Triangles

Consider the number triangle shown below. Write a program that calculates the highest sum of numbers that can be passed on a route that starts at the top and ends somewhere on the base. Each step can go either diagonally down to the left or diagonally down to the right.

```
          7
        3   8
      8   1   0
    2   7   4   4
  4   5   2   6   5
```

In the sample above, the route from 7 to 3 to 8 to 7 to 5 produces the highest sum: 30.

PROGRAM NAME: numtri

INPUT FORMAT

The first line contains R (1 <= R <= 1000), the number of rows. Each subsequent line contains the integers for that particular row of the triangle. All the supplied integers are non-negative and no larger than 100.

SAMPLE INPUT (file numtri.in)

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

OUTPUT FORMAT

A single line containing the largest sum using the traversal specified.

SAMPLE OUTPUT (file numtri.out)

```
30
```

1.5.2 Prime Palindromes

The number 151 is a prime palindrome because it is both a prime number and a palindrome (it is the same number when

read forward as backward). Write a program that finds all prime palindromes in the range of two supplied numbers a and b (5 <= a < b <= 100,000,000); both a and b are considered to be within the range .

PROGRAM NAME: pprime

INPUT FORMAT

Line 1: Two integers, a and b

SAMPLE INPUT (file pprime.in)

```
5 500
```

OUTPUT FORMAT

The list of palindromic primes in numerical order, one per line.

SAMPLE OUTPUT (file pprime.out)

```
5
7
11
101
131
151
181
191
313
353
373
383
```

1.5.3 Superprime Rib

Butchering Farmer John's cows always yields the best prime rib. You can tell prime ribs by looking at the digits lovingly stamped across them, one by one, by FJ and the USDA. Farmer John ensures that a purchaser of his prime ribs gets really prime ribs because when sliced from the right, the numbers on the ribs continue to stay prime right down to the last rib, e.g.:

```
    7  3  3  1
```

The set of ribs denoted by 7331 is prime; the three ribs 733 are prime; the two ribs 73 are prime, and, of course, the last rib, 7, is prime. The number 7331 is called a superprime of length 4.

Write a program that accepts a number N 1 <=N<=8 of ribs and prints all the superprimes of that length.

The number 1 (by itself) is not a prime number.

PROGRAM NAME: sprime

INPUT FORMAT

A single line with the number N.

SAMPLE INPUT (file sprime.in)

```
4
```

OUTPUT FORMAT

The superprime ribs of length N, printed in ascending order one per line.

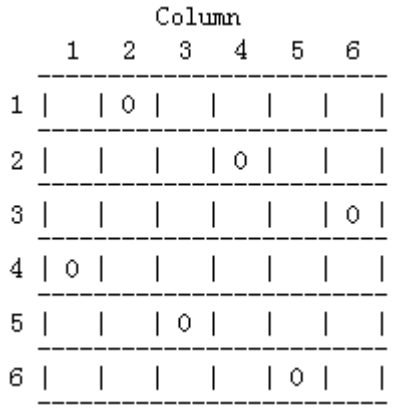
SAMPLE OUTPUT (file sprime.out)

```
2333
2339
2393
2399
2939
3119
3137
3733
3739
3793
```

```
3797
5939
7193
7331
7333
7393
```

1.5.4 Checker Challenge

Examine the 6x6 checkerboard below and note that the six checkers are arranged on the board so that one and only one is placed in each row and each column, and there is never more than one in any diagonal. (Diagonals run from southeast to northwest and southwest to northeast and include all diagonals, not just the major two.)



The solution shown above is described by the sequence 2 4 6 1 3 5, which gives the column positions of the checkers for each row from 1 to 6:

ROW	1	2	3	4	5	6
COLUMN	2	4	6	1	3	5

This is one solution to the checker challenge. Write a program that finds all unique solution sequences to the Checker Challenge (with ever growing values of N). Print the solutions using the column notation described above. Print the the first three solutions in numerical order, as if the checker positions form the digits of a large number, and then a line with the total number of solutions.

Special note: the larger values of N require your program to be especially efficient. Do not precalculate the value and print it (or even find a formula for it); that's cheating. Work on your program until it can solve the problem properly. If you insist on cheating, your login to the USACO training pages will be removed and you will be disqualified from all USACO competitions. YOU HAVE BEEN WARNED.

TIME LIMIT: 1 CPU second

PROGRAM NAME: checker

INPUT FORMAT

A single line that contains a single integer N (6 <= N <= 13) that is the dimension of the N x N checkerboard.

SAMPLE INPUT (file checker.in)

```
6
```

OUTPUT FORMAT

The first three lines show the first three solutions found, presented as N numbers with a single space between them. The fourth line shows the total number of solutions found.

SAMPLE OUTPUT (file checker.out)

```
2 4 6 1 3 5
3 6 2 5 1 4
4 1 5 2 6 3
4
```

Chapter 2 Bigger Challenges

Section 2.1

2.1.1 The Castle

IOI'94 - Day 1

In a stroke of luck almost beyond imagination, Farmer John was sent a ticket to the Irish Sweepstakes (really a lottery) for his birthday. This ticket turned out to have only the winning number for the lottery! Farmer John won a fabulous castle in the Irish countryside.

Bragging rights being what they are in Wisconsin, Farmer John wished to tell his cows all about the castle. He wanted to know how many rooms it has and how big the largest room was. In fact, he wants to take out a single wall to make an even bigger room.

Your task is to help Farmer John know the exact room count and sizes.

The castle floorplan is divided into M (wide) by N (1 <=M,N<=50) square modules. Each such module can have between zero and four walls. Castles always have walls on their "outer edges" to keep out the wind and rain.

Consider this annotated floorplan of a castle:

```
    1   2   3   4   5   6   7
  #############################
1 #   |   #   |   #   |   |   #
  #####---#####---#---#####---#
2 #   #   |   #   #   #   #   #
  #---#####---#####---#####---#
3 #   |   |   #   #   #   #   #
  #---#########---#####---#---#
4 # ->#   |   |   |   |   #   #
  #############################

#  = Wall     -,|  = No wall
-> = Points to the wall to remove to
     make the largest possible new room
```

By way of example, this castle sits on a 7 x 4 base. A "room" includes any set of connected "squares" in the floor plan. This floorplan contains five rooms (whose sizes are 9, 7, 3, 1, and 8 in no particular order).

Removing the wall marked by the arrow merges a pair of rooms to make the largest possible room that can be made by removing a single wall.

The castle always has at least two rooms and always has a wall that can be removed.

PROGRAM NAME: castle

INPUT FORMAT

The map is stored in the form of numbers, one number for each module, M numbers on each of N lines to describe the floorplan. The input order corresponds to the numbering in the example diagram above.

Each module number tells how many of the four walls exist and is the sum of up to four integers:

- 1: wall to the west
- 2: wall to the north
- 4: wall to the east
- 8: wall to the south

Inner walls are defined twice; a wall to the south in module 1,1 is also indicated as a wall to the north in module 2,1.

Line 1: Two space-separated integers: M and N

Line 2..: M x N integers, several per line.

SAMPLE INPUT (file castle.in)

```
7 4
11 6 11 6 3 10 6
7 9 6 13 5 15 5
1 10 12 7 13 7 5
13 11 10 8 10 12 13
```

OUTPUT FORMAT

The output contains several lines:

Line 1: The number of rooms the castle has.

Line 2: The size of the largest room

Line 3: The size of the largest room creatable by removing one wall

Line 4: The single wall to remove to make the largest room possible

Choose the optimal wall to remove from the set of optimal walls by choosing the module farthest to the west (and then, if still tied, farthest to the south). If still tied, choose 'N' before 'E'. Name that wall by naming the module that borders it on either the west or south, along with a direction of N or E giving the location of the wall with respect to the module.

SAMPLE OUTPUT (file castle.out)

```
5
9
16
4 1 E
```

2.1.2 Ordered Fractions

Consider the set of all reduced fractions between 0 and 1 inclusive with denominators less than or equal to N.

Here is the set when N = 5:

```
0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5 1/1
```

Write a program that, given an integer N between 1 and 160 inclusive, prints the fractions in order of increasing magnitude.

PROGRAM NAME: frac1

INPUT FORMAT

One line with a single integer N.

SAMPLE INPUT (file frac1.in)

```
5
```

OUTPUT FORMAT

One fraction per line, sorted in order of magnitude.

SAMPLE OUTPUT (file frac1.out)

```
0/1
1/5
1/4
1/3
2/5
1/2
3/5
2/3
3/4
4/5
1/1
```

2.1.3 Sorting a Three-Valued Sequence

IOI'96 - Day 2

Sorting is one of the most frequently performed computational tasks. Consider the special sorting problem in which the records to be sorted have at most *three* different key values. This happens for instance when we sort medalists of a competition according to medal value, that is, gold medalists come first, followed by silver, and bronze medalists come last.

In this task the possible key values are the integers 1, 2 and 3. The required sorting order is non-decreasing. However, sorting has to be accomplished by a sequence of exchange operations. An exchange operation, defined by two position numbers p and q, exchanges the elements in positions p and q.

You are given a sequence of key values. Write a program that computes the minimal number of exchange operations that are necessary to make the sequence sorted.

PROGRAM NAME: sort3

INPUT FORMAT

Line 1: N (1 <= N <= 1000), the number of records to be sorted

Lines 2-N+1: A single integer from the set {1, 2, 3}

SAMPLE INPUT (file sort3.in)

```
9
2
2
1
3
3
3
2
3
1
```

OUTPUT FORMAT

A single line containing the number of exchanges required

SAMPLE OUTPUT (file sort3.out)

```
4
```

2.1.4 Healthy Holsteins

Burch & Kolstad

Farmer John prides himself on having the healthiest dairy cows in the world. He knows the vitamin content for one scoop of each feed type and the minimum daily vitamin requirement for the cows. Help Farmer John feed his cows so they stay healthy while minimizing the number of scoops that a cow is fed.

Given the daily requirements of each kind of vitamin that a cow needs, identify the smallest combination of scoops of feed a cow can be fed in order to meet at least the minimum vitamin requirements.

Vitamins are measured in integer units. Cows can be fed at most one scoop of any feed type. It is guaranteed that a solution exists for all contest input data.

PROGRAM NAME: holstein

INPUT FORMAT

Line 1: integer V (1 <= V <= 25), the number of types of vitamins

Line 2: V integers (1 <= each one <= 1000), the minimum requirement for each of the V vitamins that a cow requires each day

Line 3: integer G (1 <= G <= 15), the number of types of feeds available

Lines 4..G+3: V integers (0 <= each one <= 1000), the amount of each vitamin that one scoop of this feed contains. The first line of these G lines describes feed #1; the second line describes feed #2; and so on.

SAMPLE INPUT (file holstein.in)

```
4
100 200 300 400
3
50   50  50  50
200 300 200 300
900 150 389 399
```

OUTPUT FORMAT

The output is a single line of output that contains:

- the minimum number of scoops a cow must eat, followed by:
- a SORTED list (from smallest to largest) of the feed types the cow is given

If more than one set of feedtypes yield a minimum of scoops, choose the set with the smallest feedtype numbers.

SAMPLE OUTPUT (file holstein.out)

```
2 1 3
```

2.1.5 Hamming Codes

Rob Kolstad

Given N, B, and D: Find a set of N codewords (1 <= N <= 64), each of length B bits (1 <= B <= 8), such that each of the codewords is at least Hamming distance of D (1 <= D <= 7) away from each of the other codewords. The Hamming distance between a pair of codewords is the number of binary bits that differ in their binary notation. Consider the two codewords 0x554 and 0x234 and their differences (0x554 means the hexadecimal number with hex digits 5, 5, and 4):

```
       0x554 = 0101 0101 0100
       0x234 = 0010 0011 0100
Bit differences: xxx  xx
```

Since five bits were different, the Hamming distance is 5.

PROGRAM NAME: hamming

INPUT FORMAT

N, B, D on a single line

SAMPLE INPUT (file hamming.in)

```
16 7 3
```

OUTPUT FORMAT

N codewords, sorted, in decimal, ten per line. In the case of multiple solutions, your program should output the solution which, if interpreted as a base 2^B integer, would have the least value.

SAMPLE OUTPUT (file hamming.out)

```
0 7 25 30 42 45 51 52 75 76
82 85 97 102 120 127
```

Section 2.2

2.2.1 Preface Numbering

A certain book's prefaces are numbered in upper case Roman numerals. Traditional Roman numeral values use a single letter to represent a certain subset of decimal numbers. Here is the standard set:

```
I   1     L   50    M  1000
V   5     C  100
X  10     D  500
```

As many as three of the same marks that represent 10^n may be placed consecutively to form other numbers:

- III is 3
- CCC is 300

Marks that have the value $5x10^n$ are never used consecutively.

Generally (with the exception of the next rule), marks are connected together and written in descending order to form even more numbers:

- • CCLXVIII = 100+100+50+10+5+1+1+1 = 268

Sometimes, a mark that represents 10^n is placed before a mark of one of the two next higher values (I before V or X; X before L or C; etc.). In this case, the value of the smaller mark is SUBTRACTED from the mark it precedes:

- IV = 4
- IX = 9
- XL = 40

This compound mark forms a unit and may not be combined to make another compound mark (e.g., IXL is wrong for 39; XXXIX is correct).

Compound marks like XD, IC, and XM are not legal, since the smaller mark is too much smaller than the larger one. For XD (wrong for 490), one would use CDXC; for IC (wrong for 99), one would use XCIX; for XM (wrong for 990), one would use CMXC. 90 is expressed XC and not LXL, since L followed by X connotes that successive marks are X or smaller (probably, anyway).

Given N (1 <= N < 3,500), the number of pages in the preface of a book, calculate and print the number of I's, V's, etc. (in order from lowest to highest) required to typeset all the page numbers (in Roman numerals) from 1 through N. Do not print letters that do not appear in the page numbers specified.

If N = 5, then the page numbers are: I, II, III, IV, V. The total number of I's is 7 and the total number of V's is 2.

PROGRAM NAME: preface

INPUT FORMAT

A single line containing the integer N.

SAMPLE INPUT (file preface.in)

```
5
```

OUTPUT FORMAT

The output lines specify, in ascending order of Roman numeral letters, the letter, a single space, and the number of times that letter appears on preface page numbers. Stop printing letter totals after printing the highest value letter used to form preface numbers in the specified set.

SAMPLE OUTPUT (file preface.out)

```
I 7
V 2
```

2.2.2 Subset Sums

JRM

For many sets of consecutive integers from 1 through N (1 <= N <= 39), one can partition the set into two sets whose sums are identical.

For example, if N=3, one can partition the set {1, 2, 3} in one way so that the sums of both subsets are identical:

- {3} and {1,2}

This counts as a single partitioning (i.e., reversing the order counts as the same partitioning and thus does not increase the count of partitions).

If N=7, there are four ways to partition the set {1, 2, 3, ... 7} so that each partition has the same sum:

- {1,6,7} and {2,3,4,5}
- {2,5,7} and {1,3,4,6}
- {3,4,7} and {1,2,5,6}
- {1,2,4,7} and {3,5,6}

Given N, your program should print the number of ways a set containing the integers from 1 through N can be partitioned into two sets whose sums are identical. Print 0 if there are no such ways.

Your program must calculate the answer, not look it up from a table.

PROGRAM NAME: subset

INPUT FORMAT

The input file contains a single line with a single integer representing N, as above.

SAMPLE INPUT (file subset.in)

```
7
```

OUTPUT FORMAT

The output file contains a single line with a single integer that tells how many same-sum partitions can be made from the set {1, 2, ..., N}. The output file should contain 0 if there are no ways to make a same-sum partition.

SAMPLE OUTPUT (file subset.out)

```
4
```

2.2.3 Runaround Numbers

Runaround numbers are integers with unique digits, none of which is zero (e.g., 81362) that also have an interesting property, exemplified by this demonstration:

- If you start at the left digit (8 in our number) and count that number of digits to the right (wrapping back to the first digit when no digits on the right are available), you'll end up at a new digit (a number which does not end up at a new digit is not a Runaround Number). Consider: 8 1 3 6 2 which cycles through eight digits: 1 3 6 2 8 1 3 6 so the next digit is 6.
- Repeat this cycle (this time for the six counts designed by the `6') and you should end on a new digit: 2 8 1 3 6 2, namely 2.
- Repeat again (two digits this time): 8 1
- Continue again (one digit this time): 3
- One more time: 6 2 8 and you have ended up back where you started, after touching each digit once. If you don't end up back where you started after touching each digit once, your number is not a Runaround number.

Given a number M (that has anywhere from 1 through 9 digits), find and print the next runaround number higher than M, which will always fit into an unsigned long integer for the given test data.

PROGRAM NAME: runround

INPUT FORMAT

A single line with a single integer, M

SAMPLE INPUT (file runround.in)

```
81361
```

OUTPUT FORMAT

A single line containing the next runaround number higher than the input value, M.

SAMPLE OUTPUT (file runround.out)

```
81362
```

2.2.4 Party Lamps

IOI 98

To brighten up the gala dinner of the IOI'98 we have a set of **N** (10 <= N <= 100) colored lamps numbered from 1 to **N**.

The lamps are connected to four buttons:

- Button 1: When this button is pressed, all the lamps change their state: those that are ON are turned OFF and those that are OFF are turned ON.
- Button 2: Changes the state of all the odd numbered lamps.
- Button 3: Changes the state of all the even numbered lamps.
- Button 4: Changes the state of the lamps whose number is of the form 3xK+1 (with K>=0), i.e., 1,4,7,...

A counter C records the total number of button presses.

When the party starts, all the lamps are ON and the counter C is set to zero.

You are given the value of counter C (0 <= C <= 10000) and the final state of some of the lamps after some operations have been executed. Write a program to determine all the possible final configurations of the N lamps that are consistent with the given information, without repetitions.

PROGRAM NAME: lamps

INPUT FORMAT

Line 1: N

Line 2: Final value of C

Line 3: Some lamp numbers ON in the final configuration, separated by one space and terminated by the integer -1.

Line 4: Some lamp numbers OFF in the final configuration, separated by one space and terminated by the integer -1. No lamp will be listed twice in the input.

SAMPLE INPUT (file lamps.in)

```
10
1
-1
7 -1
```

In this case, there are 10 lamps and only one button has been pressed. Lamp 7 is OFF in the final configuration.

OUTPUT FORMAT

Lines with all the possible final configurations (without repetitions) of all the lamps. Each line has N characters, where the first character represents the state of lamp 1 and the last character represents the state of lamp N. A 0 (zero) stands for a lamp that is OFF, and a 1 (one) stands for a lamp that is ON. The lines must be ordered from least to largest (as binary numbers).

If there are no possible configurations, output a single line with the single word `IMPOSSIBLE'

SAMPLE OUTPUT (file lamps.out)

```
0000000000
0101010101
0110110110
```

In this case, there are three possible final configurations:

- All lamps are OFF
- Lamps 1, 4, 7, 10 are OFF and lamps 2, 3, 5, 6, 8, 9 are **ON**.
- Lamps 1, 3, 5, 7, 9 are OFF and lamps 2, 4, 6, 8, 10 are **ON**.

Section 2.3

2.3.1 Longest Prefix

IOI'96

The structure of some biological objects is represented by the sequence of their constituents, where each part is denote by an uppercase letter. Biologists are interested in decomposing a long sequence into shorter ones called *primitives*.

We say that a sequence S can be composed from a given set of primitives P if there is a some sequence of (possibly repeated) primitives from the set whose concatenation equals S. Not necessarily all primitives need be present. For instance the sequence ABABACABAABcan be composed from the set of primitives

```
{A, AB, BA, CA, BBC}
```

The first K characters of S are the *prefix of S with length K*. Write a program which accepts as input a set of primitives and a sequence of constituents and then computes the length of the longest prefix that can be composed from primitives.

PROGRAM NAME: prefix

INPUT FORMAT

First, the input file contains the list (length 1..200) of primitives (length 1..10) expressed as a series of space-separated strings of upper-case characters on one or more lines. The list of primitives is terminated by a line that contains nothing more than a period (`.'). No primitive appears twice in the list. Then, the input file contains a sequence S (length 1..200,000) expressed as one or more lines, none of which exceeds 76 letters in length. The "newlines" (line terminators) are not part of the string S.

SAMPLE INPUT (file prefix.in)

```
A AB BA CA BBC
.
ABABACABAABC
```

OUTPUT FORMAT

A single line containing an integer that is the length of the longest prefix that can be composed from the set P.

SAMPLE OUTPUT (file prefix.out)

```
11
```

2.3.2 Cow Pedigrees

Silviu Ganceanu -- 2003

Farmer John is considering purchasing a new herd of cows. In this new herd, each mother cow gives birth to two children. The relationships among the cows can easily be represented by one or more binary trees with a total of N (3 <= N < 200) nodes. The trees have these properties:

- The degree of each node is 0 or 2. The degree is the count of the node's immediate children.
- The height of the tree is equal to K (1 < K <100). The height is the number of nodes on the longest path from the root to any leaf; a leaf is a node with no children.

How many different possible pedigree structures are there? A pedigree is different if its tree structure differs from that of another pedigree. Output the remainder when the total number of different possible pedigrees is divided by 9901.

PROGRAM NAME: nocows

INPUT FORMAT

- Line 1: Two space-separated integers, N and K.

SAMPLE INPUT (file nocows.in)

```
5 3
```

OUTPUT FORMAT

- Line 1: One single integer number representing the number of possible pedigrees MODULO 9901.

SAMPLE OUTPUT (file nocows.out)

```
2
```

OUTPUT DETAILS

Two possible pedigrees have 5 nodes and height equal to 3:

```
      @                   @
     / \                 / \
    @   @      and      @   @
   / \                     / \
  @   @                   @   @
```

2.3.3 Zero Sum

Consider the sequence of digits from 1 through N (where N=9) in increasing order: `1 2 3 ... N`.

Now insert either a `+' for addition or a `-' for subtraction or a ` ' [blank] to run the digits together between each pair of digits (not in front of the first digit). Calculate the result that of the expression and see if you get zero.

Write a program that will find all sequences of length N that produce a zero sum.

PROGRAM NAME: zerosum

INPUT FORMAT

A single line with the integer N (3 <= N <= 9).

SAMPLE INPUT (file zerosum.in)

```
7
```

OUTPUT FORMAT

In ASCII order, show each sequence that can create 0 sum with a `+', `-', or ` ' between each pair of numbers.

SAMPLE OUTPUT (file zerosum.out)

```
1+2-3+4-5-6+7
1+2-3-4+5+6-7
1-2 3+4+5+6+7
1-2 3-4 5+6 7
1-2+3+4-5+6-7
1-2-3-4-5+6+7
```

2.3.4 Money Systems

The cows have not only created their own government but they have chosen to create their own money system. In their own rebellious way, they are curious about values of coinage. Traditionally, coins come in values like 1, 5, 10, 20 or 25, 50, and 100 units, sometimes with a 2 unit coin thrown in for good measure.

The cows want to know how many different ways it is possible to dispense a certain amount of money using various coin systems. For instance, using a system of {1, 2, 5, 10, ...} it is possible to create 18 units several different ways, including: 18x1, 9x2, 8x2+2x1, 3x5+2+1, and many others.

Write a program to compute how many ways to construct a given amount of money using supplied coinage. It is guaranteed that the total will fit into both a signed long long (C/C++) and Int64 (Free Pascal).

PROGRAM NAME: money

INPUT FORMAT

The number of coins in the system is V (1 <= V <= 25).

The amount money to construct is N (1 <= N <= 10,000).

Line 1: Two integers, V and N

Lines 2..: V integers that represent the available coins (no particular number of integers per line)

SAMPLE INPUT (file money.in)

```
3 10
1 2 5
```

OUTPUT FORMAT

A single line containing the total number of ways to construct N money units using V coins.

SAMPLE OUTPUT (file money.out)

```
10
```

2.3.5 Controlling Companies

Some companies are partial owners of other companies because they have acquired part of their total shares of stock. For example, Ford owns 12% of Mazda. It is said that a company A controls company B if at least one of the following conditions is satisfied:

- Company A = Company B
- Company A owns more than 50% of Company B
- Company A controls K (K >= 1) companies denoted C_1, ..., C_K with each company C_i owning x_i% of company B and $x_1 + ... + x_K > 50\%$.

Given a list of triples (i,j,p) which denote company i owning p% of company j, calculate all the pairs (h,s) in which company h controls company s. There are at most 100 companies.

Write a program to read the list of triples (i,j,p) where i, j and p are positive integers all in the range (1..100) and find all the pairs (h,s) so that company h controls company s.

PROGRAM NAME: concom

INPUT FORMAT

Line 1: n, the number of input triples to follow

Line 2..n+1: Three integers per line as a triple (i,j,p) described above.

SAMPLE INPUT (file concom.in)

```
3
1 2 80
2 3 80
3 1 20
```

OUTPUT FORMAT

List 0 or more companies that control other companies. Each line contains two integers that denote that the company whose number is the first integer controls the company whose number is the second integer. Order the lines in ascending order of the first integer (and ascending order of the second integer to break ties). Do not print that a company controls itself.

SAMPLE OUTPUT (file concom.out)

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

Section 2.4

2.4.1 The Tamworth Two

BIO '98 - Richard Forster

A pair of cows is loose somewhere in the forest. Farmer John is lending his expertise to their capture. Your task is to model their behavior.

The chase takes place on a 10 by 10 planar grid. Squares can be empty or they can contain:

- an obstacle,
- the cows (who always travel together), or
- Farmer John.

The cows and Farmer John can occupy the same square (when they `meet') but neither the cows nor Farmer John can share a square with an obstacle.

```
Each square is
represented
as follows:

. Empty square
* Obstacle
C Cows
F Farmer
```

```
Here is a sample grid:
*...*.....
......*...
...*...*..
..........
...*.F....
*.....*...
...*......
..C......*
...*.*....
.*.*......
```

The cows wander around the grid in a fixed way. Each minute, they either move forward or rotate. Normally, they move one square in the direction they are facing. If there is an obstacle in the way or they would leave the board by walking `forward', then they spend the entire minute rotating 90 degrees clockwise.

Farmer John, wise in the ways of cows, moves in exactly the same way.

The farmer and the cows can be considered to move simultaneously during each minute. If the farmer and the cows pass each other while moving, they are not considered to have met. The chase ends when Farmer John and the cows occupy the same square at the end of a minute.

Read a ten-line grid that represents the initial state of the cows, Farmer John, and obstacles. Each of the ten lines contains exactly ten characters using the coding above. There is guaranteed to be only one farmer and one pair of cows. The cows and Farmer John will not initially be on the same square.

Calculate the number of minutes until the cows and Farmer John meet. Assume both the cows and farmer begin the simulation facing in the `north' direction. Print 0 if they will never meet.

PROGRAM NAME: ttwo

INPUT FORMAT

Lines 1-10: Ten lines of ten characters each, as explained above

SAMPLE INPUT (file ttwo.in)

```
*...*.....
......*...
...*...*..
..........
...*.F....
*.....*...
...*......
..C......*
...*.*....
.*.*......
```

OUTPUT FORMAT

A single line with the integer number of minutes until Farmer John and the cows meet. Print 0 if they will never meet.

SAMPLE OUTPUT (file ttwo.out)

```
49
```

2.4.2 Overfencing

Kolstad and Schrijvers

Farmer John went crazy and created a huge maze of fences out in a field. Happily, he left out two fence segments on the edges, and thus created two "exits" for the maze. Even more happily, the maze he created by this overfencing experience is a `perfect' maze: you can find a way out of the maze from any point inside it.

Given W (1 <= W <= 38), the width of the maze; H (1 <= H <= 100), the height of the maze; 2*H+1 lines with width 2*W+1 characters that represent the maze in a format like that shown later - then calculate the number of steps required to exit the maze from the `worst' point in the maze (the point that is `farther' from either exit even when walking optimally to the closest exit). Of course, cows walk only parallel or perpendicular to the x-y axes; they do not walk on a diagonal. Each move to a new square counts as a single unit of distance (including the move "out" of the maze.

Here's what one particular W=5, H=3 maze looks like:

```
+-+-+-+-+-+
|         |
+-+ +-+ + +
|     | | |
+ +-+-+ + +
| |     |
+-+ +-+-+-+
```

Fenceposts appear only in odd numbered rows and and odd numbered columns (as in the example). The format should be obvious and self explanatory. Each maze has exactly two blank walls on the outside for exiting.

PROGRAM NAME: maze1

INPUT FORMAT

Line 1: W and H, space separated

Lines 2 through 2*H+2: 2*W+1 characters that represent the maze

SAMPLE INPUT (file maze1.in)

```
5 3
+-+-+-+-+-+
|         |
+-+ +-+ + +
|     | | |
+ +-+-+ + +
| |     |
+-+ +-+-+-+
```

OUTPUT FORMAT

A single integer on a single output line. The integer specifies the minimal number of steps that guarantee a cow can exit the maze from any possible point inside the maze.

SAMPLE OUTPUT (file maze1.out)

```
9
```

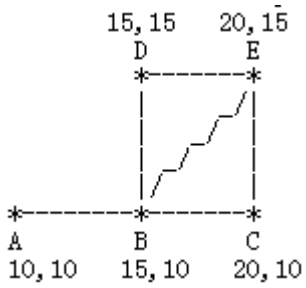
The lower left-hand corner is *nine* steps from the closest exit.

2.4.3 Cow Tours

Farmer John has a number of pastures on his farm. Cow paths connect some pastures with certain other pastures, forming a field. But, at the present time, you can find at least two pastures that cannot be connected by any sequence of cow paths, thus partitioning Farmer John's farm into multiple fields.

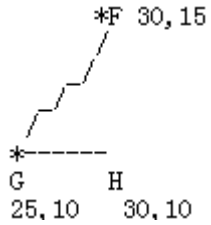
Farmer John would like add a single a cow path between one pair of pastures using the constraints below.

A field's `diameter' is defined to be the largest distance of all the shortest walks between any pair of pastures in the field. Consider the field below with five pastures, located at the points shown, and cow paths marked by lines:



The `diameter' of this field is approximately 12.07106, since the longest of the set of shortest paths between pairs of pastures is the path from A to E (which includes the point set {A,B,E}). No other pair of pastures in this field is farther apart when connected by an optimal sequence of cow paths.

Suppose another field on the same plane is connected by cow paths as follows:



In the scenario of just two fields on his farm, Farmer John would add a cow path between a point in each of these two fields (namely point sets {A,B,C,D,E} and {F,G,H}) so that the joined set of pastures {A,B,C,D,E,F,G,H} has the smallest possible diameter.

Note that cow paths do not connect just because they cross each other; they only connect at listed points.

The input contains the pastures, their locations, and a symmetric "adjacency" matrix that tells whether pastures are connected by cow paths. Pastures are not considered to be connected to themselves. Here's one annotated adjacency list for the pasture {A,B,C,D,E,F,G,H} as shown above:

```
  A B C D E F G H
A 0 1 0 0 0 0 0 0
B 1 0 1 1 1 0 0 0
C 0 1 0 0 1 0 0 0
D 0 1 0 0 1 0 0 0
E 0 1 1 1 0 0 0 0
F 0 0 0 0 0 0 1 0
G 0 0 0 0 0 1 0 1
H 0 0 0 0 0 0 1 0
```

Other equivalent adjacency lists might permute the rows and columns by using some order other than alphabetical to show the point connections. The input data contains no names for the points.

The input will contain at least two pastures that are not connected by any sequence of cow paths.

Find a way to connect exactly two pastures in the input with a cow path so that the new combined field has the smallest possible diameter of any possible pair of connected pastures. Output that smallest possible diameter.

PROGRAM NAME: cowtour

INPUT FORMAT

Line 1:	An integer, N (1 <= N <= 150), the number of pastures
Line 2-N+1:	Two integers, X and Y (0 <= X ,Y<= 100000), that denote that X,Y grid location of the pastures; all input pastures are unique.
Line N+2-2*N+1:	lines, each containing N digits (0 or 1) that represent the adjacency matrix as described above, where the rows' and columns' indices are in order of the points just listed.

SAMPLE INPUT (file cowtour.in)

```
8
10 10
15 10
20 10
15 15
20 15
30 15
25 10
30 10
01000000
10111000
01001000
01001000
01110000
00000010
00000101
00000010
```

OUTPUT FORMAT

The output consists of a single line with the diameter of the newly joined pastures. Print the answer to exactly six decimal places. Do not perform any special rounding on your output.

SAMPLE OUTPUT (file cowtour.out)

```
22.071068
```

2.4.4 Bessie Come Home

Kolstad & Burch

It's dinner time, and the cows are out in their separate pastures. Farmer John rings the bell so they will start walking to the barn. Your job is to figure out which one cow gets to the barn first (the supplied test data will always have exactly one fastest cow).

Between milkings, each cow is located in her own pasture, though some pastures have no cows in them. Each pasture is connected by a path to one or more other pastures (potentially including itself). Sometimes, two (potentially self-same) pastures are connected by more than one path. One or more of the pastures has a path to the barn. Thus, all cows have a path to the barn and they always know the shortest path. Of course, cows can go either direction on a path and they all walk at the same speed.

The pastures are labeled `a'..`z' and `A'..`Y'. One cow is in each pasture labeled with a capital letter. No cow is in a pasture labeled with a lower case letter. The barn's label is `Z'; no cows are in the barn, though.

PROGRAM NAME: comehome

INPUT FORMAT

Line 1:	Integer P (1 <= P <= 10000) the number of paths that interconnect the pastures (and the barn)
Line 2..P+1:	Space separated, two letters and an integer: the names of the interconnected pastures/barn and the distance between them (1 <= distance <= 1000)

SAMPLE INPUT (file comehome.in)

```
5
A d 6
B d 3
C e 9
d Z 8
e Z 3
```

OUTPUT FORMAT

A single line containing two items: the capital letter name of the pasture of the cow that arrives first back at the barn, the length of the path followed by that cow.

SAMPLE OUTPUT (file comehome.out)

```
B 11
```

2.4.5 Fractions to Decimals

Write a program that will accept a fraction of the form N/D, where N is the numerator and D is the denominator and print the decimal representation. If the decimal representation has a repeating sequence of digits, indicate the sequence by enclosing it in brackets. For example, 1/3 = .33333333...is denoted as 0.(3), and 41/333 = 0.123123123...is denoted as 0.(123). Use xxx.0 to denote an integer. Typical conversions are:

```
1/3     =  0.(3)
22/5    =  4.4
1/7     =  0.(142857)
2/2     =  1.0
3/8     =  0.375
45/56   =  0.803(571428)
```

PROGRAM NAME: fracdec

INPUT FORMAT

A single line with two space separated integers, N and D, 1 <= N,D <= 100000.

SAMPLE INPUT (file fracdec.in)

```
45 56
```

OUTPUT FORMAT

The decimal expansion, as detailed above. If the expansion exceeds 76 characters in length, print it on multiple lines with 76 characters per line.

SAMPLE OUTPUT (file fracdec.out)

```
0.803(571428)
```

Chapter 3 Techniques more subtle

Section 3.1

3.1.1 Agri-Net

Russ Cox

Farmer John has been elected mayor of his town! One of his campaign promises was to bring internet connectivity to all farms in the area. He needs your help, of course.

Farmer John ordered a high speed connection for his farm and is going to share his connectivity with the other farmers. To minimize cost, he wants to lay the minimum amount of optical fiber to connect his farm to all the other farms.

Given a list of how much fiber it takes to connect each pair of farms, you must find the minimum amount of fiber needed to connect them all together. Each farm must connect to some other farm such that a packet can flow from any one farm to any other farm.

The distance between any two farms will not exceed 100,000.

PROGRAM NAME: agrinet

INPUT FORMAT

Line 1: The number of farms, N (3 <= N <= 100).

Line 2..end: The subsequent lines contain the N x N connectivity matrix, where each element shows the distance from on farm to another. Logically, they are N lines of N space-separated integers. Physically, they are limited in length to 80 characters, so some lines continue onto others. Of course, the diagonal will be 0, since the distance from farm i to itself is not interesting for this problem.

SAMPLE INPUT (file agrinet.in)

```
4
0 4 9 21
4 0 8 17
9 8 0 16
21 17 16 0
```

OUTPUT FORMAT

The single output contains the integer length that is the sum of the minimum length of fiber required to connect the entire set of farms.

SAMPLE OUTPUT (file agrinet.out)

```
28
```

3.1.2 Score Inflation

The more points students score in our contests, the happier we here at the USACO are. We try to design our contests so that people can score as many points as possible, and would like your assistance.

We have several categories from which problems can be chosen, where a "category" is an unlimited set of contest problems which all require the same amount of time to solve and deserve the same number of points for a correct solution. Your task is write a program which tells the USACO staff how many problems from each category to include in a contest so as to maximize the total number of points in the chosen problems while keeping the total solution time within the length of the contest.

The input includes the length of the contest, M (1 <= M <= 10,000) (don't worry, you won't have to compete in the longer contests until training camp) and N, the number of problem categories, where 1 <= N <= 10,000.

Each of the subsequent N lines contains two integers describing a category: the first integer tells the number of points a problem from that category is worth (1 <= points <= 10000); the second tells the number of minutes a problem from that category takes to solve (1 <= minutes <= 10000).

Your program should determine the number of problems we should take from each category to make the highest-scoring contest solvable within the length of the contest. Remember, the number from any category can be any nonnegative integer (0, one, or many). Calculate the maximum number of possible points.

PROGRAM NAME: inflate

INPUT FORMAT

Line 1: M, N -- contest minutes and number of problem classes

Lines 2-N+1: Two integers: the points and minutes for each class

SAMPLE INPUT (file inflate.in)

```
300 4
100 60
250 120
120 100
35 20
```

OUTPUT FORMAT

A single line with the maximum number of points possible given the constraints.

SAMPLE OUTPUT (file inflate.out)

```
605
```

(Take two problems from #2 and three from #4.)

3.1.3 Humble Numbers

For a given set of K prime numbers S = {p_1, p_2, ..., p_K}, consider the set of all numbers whose prime factors are a subset of S. This set contains, for example, p_1, p_1p_2, p_1p_1, and $p_1p_2p_3$ (among others). This is the set of `humble numbers' for the input set S. Note: The number 1 is explicitly declared not to be a humble number.

Your job is to find the Nth humble number for a given set S. Long integers (signed 32-bit) will be adequate for all solutions.

PROGRAM NAME: humble

INPUT FORMAT

Line 1: Two space separated integers: K and N, 1 <= K <=100 and 1 <= N <= 100,000.

Line 2: K space separated positive integers that comprise the set S.

SAMPLE INPUT (file humble.in)

```
4 19
2 3 5 7
```

OUTPUT FORMAT

The Nth humble number from set S printed alone on a line.

SAMPLE OUTPUT (file humble.out)

```
27
```

3.1.4 Shaping Regions

N opaque rectangles (1 <= N <= 1000) of various colors are placed on a white sheet of paper whose size is A wide by B long. The rectangles are put with their sides parallel to the sheet's borders. All rectangles fall within the borders of the sheet so that different figures of different colors will be seen.

The coordinate system has its origin (0,0) at the sheet's lower left corner with axes parallel to the sheet's borders.

PROGRAM NAME: rect1

INPUT FORMAT

The order of the input lines dictates the order of laying down the rectangles. The first input line is a rectangle "on the bottom".

Line 1: A, B, and N, space separated (1 <= A,B <= 10,000)

Lines 2-N+1: Five integers: llx, lly, urx, ury, color: the lower left coordinates and upper right coordinates of the rectangle whose color is `color' (1 <= color <= 2500) to be placed on the white sheet. The color 1 is the same color of white as the sheet upon which the rectangles are placed.

SAMPLE INPUT (file rect1.in)

```
20 20 3
2 2 18 18 2
0 8 19 19 3
8 0 10 19 4
```

INPUT EXPLANATION

Note that the rectangle delineated by 0,0 and 2,2 is two units wide and two high. Here's a schematic diagram of the input:

```
11111111111111111111
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
33333333443333333331
11222222442222222211
11222222442222222211
11222222442222222211
11222222442222222211
11222222442222222211
11222222442222222211
11111111441111111111
11111111441111111111
```

The '4's at 8,0 to 10,19 are only two wide, not three (i.e., the grid contains a 4 and 8,0 and a 4 and 8,1 but NOT a 4 and 8,2 since this diagram can't capture what would be shown on graph paper).

OUTPUT FORMAT

The output file should contain a list of all the colors that can be seen along with the total area of each color that can be seen (even if the regions of color are disjoint), ordered by increasing color. Do not display colors with no area.

SAMPLE OUTPUT (file rect1.out)

```
1 91
2 84
3 187
4 38
```

3.1.5 Contact

IOI'98

The cows have developed a new interest in scanning the universe outside their farm with radiotelescopes. Recently, they noticed a very curious microwave pulsing emission sent right from the centre of the galaxy. They wish to know if the

emission is transmitted by some extraterrestrial form of intelligent life or if it is nothing but the usual heartbeat of the stars.

Help the cows to find the Truth by providing a tool to analyze bit patterns in the files they record. They are seeking bit patterns of length **A** through **B** inclusive (1 <= A <= B <= 12) that repeat themselves most often in each day's data file. They are looking for the patterns that repeat themselves most often. An input limit tells how many of the most frequent patterns to output.

Pattern occurrences may overlap, and only patterns that occur at least once are taken into account.

PROGRAM NAME: contact

INPUT FORMAT

Line 1: Three space-separated integers: A, B, N; (1 <= N < 50)

Lines 2 and beyond: A sequence of as many as 200,000 characters, all 0 or 1; the characters are presented 80 per line, except potentially the last line.

SAMPLE INPUT (file contact.in)

```
2 4 10
010100100100010001111011000010100110011110000100100111100100000000
```

In this example, pattern 100 occurs 12 times, and pattern 1000 occurs 5 times. The most frequent pattern is 00, with 23 occurrences.

OUTPUT FORMAT

Lines that list the N highest frequencies (in descending order of frequency) along with the patterns that occur in those frequencies. Order those patterns by shortest-to-longest and increasing binary number for those of the same frequency. If fewer than N highest frequencies are available, print only those that are.

Print the frequency alone by itself on a line. Then print the actual patterns space separated, six to a line (unless fewer than six remain).

SAMPLE OUTPUT (file contact.out)

```
23
00
15
01 10
12
100
11
11 000 001
10
010
8
0100
7
0010 1001
6
111 0000
5
011 110 1000
4
0001 0011 1100
```

3.1.6 Stamps

Given a set of N stamp values (e.g., {1 cent, 3 cents}) and an upper limit K to the number of stamps that can fit on an envelope, calculate the largest unbroken list of postages from 1 cent to M cents that can be created.

For example, consider stamps whose values are limited to 1 cent and 3 cents; you can use at most 5 stamps. It's easy to see

how to assemble postage of 1 through 5 cents (just use that many 1 cent stamps), and successive values aren't much harder:

- 6 = 3 + 3
- 7 = 3 + 3 + 1
- 8 = 3 + 3 + 1 + 1
- 9 = 3 + 3 + 3
- 10 = 3 + 3 + 3 + 1
- 11 = 3 + 3 + 3 + 1 + 1
- 12 = 3 + 3 + 3 + 3
- 13 = 3 + 3 + 3 + 3 + 1.

However, there is no way to make 14 cents of postage with 5 or fewer stamps of value 1 and 3 cents. Thus, for this set of two stamp values and a limit of K=5, the answer is M=13.

The most difficult test case for this problem has a time limit of 3 seconds.

PROGRAM NAME: stamps

INPUT FORMAT

Line 1: Two integers K and N. K (1 <= K <= 200) is the total number of stamps that can be used. N (1 <= N <= 50) is the number of stamp values.

Lines 2..end: N integers, 15 per line, listing all of the N stamp values, each of which will be at most 10000.

SAMPLE INPUT (file stamps.in)

```
5 2
1 3
```

OUTPUT FORMAT

Line 1: One integer, the number of contiguous postage values starting at 1 cent that can be formed using no more than K stamps from the set.

SAMPLE OUTPUT (file stamps.out)

```
13
```

Section 3.2

3.2.1 Factorials

The factorial of an integer N, written N!, is the product of all the integers from 1 through N inclusive. The factorial quickly becomes very large: 13! is too large to store in a 32-bit integer on most computers, and 70! is too large for most floating-point variables. Your task is to find the rightmost non-zero digit of n!. For example, 5! = 1 * 2 * 3 * 4 * 5 = 120, so the rightmost non-zero digit of 5! is 2. Likewise, 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040, so the rightmost non-zero digit of 7! is 4.

PROGRAM NAME: fact4

INPUT FORMAT

A single positive integer N no larger than 4,220.

SAMPLE INPUT (file fact4.in)

```
7
```

OUTPUT FORMAT

A single line containing but a single digit: the right most non-zero digit of N! .

SAMPLE OUTPUT (file fact4.out)

```
4
```

3.2.2 Stringsobits

Kim Schrijvers

Consider an ordered set S of strings of N (1 <= N <= 31) bits. Bits, of course, are either 0 or 1.

This set of strings is interesting because it is ordered and contains all possible strings of length N that have L (1 <= L <= N) or fewer bits that are `1'.

Your task is to read a number I (1 <= I <= sizeof(S)) from the input and print the Ith element of the ordered set for N bits with no more than L bits that are `1'.

PROGRAM NAME: kimbits

INPUT FORMAT

A single line with three space separated integers: N, L, and I.

SAMPLE INPUT (file kimbits.in)

```
5 3 19
```

OUTPUT FORMAT

A single line containing the integer that represents the Ith element from the order set, as described.

SAMPLE OUTPUT (file kimbits.out)

```
10011
```

3.2.3 Spinning Wheels

1998 ACM NE Regionals

Each of five opaque spinning wheels has one or more wedges cut out of its edges. These wedges must be aligned quickly and correctly. Each wheel also has an alignment mark (at 0 degrees) so that the wheels can all be started in a known position. Wheels rotate in the `plus degrees' direction, so that shortly after they start, they pass through 1 degree, 2 degrees, etc. (though probably not at the same time).

This is an integer problem. Wheels are never actually at 1.5 degrees or 23.51234123 degrees. For example, the wheels are considered to move instantaneously from 20 to 25 degrees during a single second or even from 30 to 40 degrees if the wheel is spinning quickly.

All angles in this problem are presumed to be integers in the range 0 <= angle <= 359. The angle of 0 degrees follows the angle of 359 degrees. Each wheel rotates at a certain integer number of degrees per second, 1 <= speed <= 180.

Wedges for each wheel are specified by an integer start angle and integer angle size (or `extent'), both specified in degrees. Wedges in the test data will be separated by at least one degree. The 'extent' also includes the original "degree" of the wedge, so '0 180' means degrees 0..180 inclusive -- one more than most would imagine.

At the start, which is time 0, all the wheels' alignment marks line up. Your program must determine the earliest time (integer seconds) at or after the start that some wedge on each wheel will align with the wedges on the other wheel so that a light beam can pass through openings on all five wedges. The wedges can align at any part of the rotation.

PROGRAM NAME: spin

INPUT FORMAT

Each of five input lines describes a wheel.

The first integer on an input line is the wheel's rotation speed. The next integer is the number of wedges, 1 <= W <= 5. The next W pairs of integers tell each wedge's start angle and extent.

SAMPLE INPUT (file spin.in)

```
30 1 0 120
50 1 150 90
60 1 60 90
```

```
70 1 180 180
90 1 180 60
```

OUTPUT FORMAT

A single line with a single integer that is the first time the wedges align so a light beam can pass through them. Print `none' (lower case, no quotes) if the wedges will never align properly.

SAMPLE OUTPUT (file spin.out)

```
9
```

3.2.4 Feed Ratios

1998 ACM Finals, Dan Adkins

Farmer John feeds his cows only the finest mixture of cow food, which has three components: Barley, Oats, and Wheat. While he knows the precise mixture of these easily mixable grains, he can not buy that mixture! He buys three other mixtures of the three grains and then combines them to form the perfect mixture.

Given a set of integer ratios barley:oats:wheat, find a way to combine them IN INTEGER MULTIPLES to form a mix with some goal ratio x:y:z.

For example, given the goal `3:4:5` and the ratios of three mixtures:

```
1:2:3
3:7:1
2:1:2
```

your program should find some minimum number of integer units (the `mixture') of the first, second, and third mixture that should be mixed together to achieve the goal ratio or print `NONE'. `Minimum number' means the sum of the three non-negative mixture integers is minimized.

For this example, you can combine eight units of mixture 1, one unit of mixture 2, and five units of mixture 3 to get seven units of the goal ratio:

```
8*(1:2:3) + 1*(3:7:1) + 5*(2:1:2) = (21:28:35) = 7*(3:4:5)
```

Integers in the goal ratio and mixture ratios are all non-negative and smaller than 100 in magnitude. The number of units of each type of feed in the mixture must be less than 100. The mixture ratios are not linear combinations of each other.

PROGRAM NAME: ratios

INPUT FORMAT

Line 1: Three space separated integers that represent the goal ratios

Line 2..4: Each contain three space separated integers that represent the ratios of the three mixtures purchased.

SAMPLE INPUT (file ratios.in)

```
3 4 5
1 2 3
3 7 1
2 1 2
```

OUTPUT FORMAT

The output file should contain one line containing four integers or the word `NONE'. The first three integers should represent the number of units of each mixture to use to obtain the goal ratio. The fourth number should be the multiple of the goal ratio obtained by mixing the initial feed using the first three integers as mixing ratios.

SAMPLE OUTPUT (file ratios.out)

```
8 1 5 7
```

3.2.5 Magic Squares

IOI'96

Following the success of the magic cube, Mr. Rubik invented its planar version, called magic squares. This is a sheet composed of 8 equal-sized squares:

```
1 2 3 4
8 7 6 5
```

In this task we consider the version where each square has a different color. Colors are denoted by the first 8 positive integers. A sheet configuration is given by the sequence of colors obtained by reading the colors of the squares starting at the upper left corner and going in clockwise direction. For instance, the configuration of *Figure 3* is given by the sequence (1,2,3,4,5,6,7,8). This configuration is the initial configuration.

Three basic transformations, identified by the letters `A', `B' and `C', can be applied to a sheet:

- 'A': exchange the top and bottom row,
- 'B': single right circular shifting of the rectangle,
- 'C': single clockwise rotation of the middle four squares.

Below is a demonstration of applying the transformations to the initial squares given above:

```
A:  8 7 6 5      B:  4 1 2 3      C:  1 7 2 4
    1 2 3 4          5 8 7 6          8 6 3 5
```

All possible configurations are available using the three basic transformations.

You are to write a program that computes a minimal sequence of basic transformations that transforms the initial configuration above to a specific target configuration.

PROGRAM NAME: msquare

INPUT FORMAT

A single line with eight space-separated integers (a permutation of (1..8)) that are the target configuration.

SAMPLE INPUT (file msquare.in)

```
2 6 8 4 5 7 3 1
```

OUTPUT FORMAT

Line 1: A single integer that is the length of the shortest transformation sequence.

Line 2: The lexically earliest string of transformations expressed as a string of characters, 60 per line except possibly the last line.

SAMPLE OUTPUT (file msquare.out)

```
7
BCABCCB
```

3.2.6 Sweet Butter

Greg Galperin -- 2001

Farmer John has discovered the secret to making the sweetest butter in all of Wisconsin: sugar. By placing a sugar cube out in the pastures, he knows the N (1 <= N <= 500) cows will lick it and thus will produce super-sweet butter which can be marketed at better prices. Of course, he spends the extra money on luxuries for the cows.

FJ is a sly farmer. Like Pavlov of old, he knows he can train the cows to go to a certain pasture when they hear a bell. He intends to put the sugar there and then ring the bell in the middle of the afternoon so that the evening's milking produces perfect milk.

FJ knows each cow spends her time in a given pasture (not necessarily alone). Given the pasture location of the cows and a description of the paths the connect the pastures, find the pasture in which to place the sugar cube so that the total distance walked by the cows when FJ rings the bell is minimized. FJ knows the fields are connected well enough that some solution is always possible.

PROGRAM NAME: butter

INPUT FORMAT

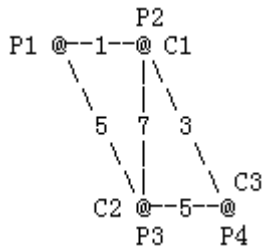
- Line 1: Three space-separated integers: N, the number of pastures: P (2 <= P <= 800), and the number of connecting paths: C (1 <= C <= 1,450). Cows are uniquely numbered 1..N. Pastures are uniquely numbered 1..P.
- Lines 2..N+1: Each line contains a single integer that is the pasture number in which a cow is grazing. Cow i's pasture is listed on line i+1.
- Lines N+2..N+C+1: Each line contains three space-separated integers that describe a single path that connects a pair of pastures and its length. Paths may be traversed in either direction. No pair of pastures is directly connected by more than one path. The first two integers are in the range 1..P; the third integer is in the range (1..225).

SAMPLE INPUT (file butter.in)

```
3 4 5
2
3
4
1 2 1
1 3 5
2 3 7
2 4 3
3 4 5
```

INPUT DETAILS

This diagram shows the connections geometrically:



OUTPUT FORMAT

- Line 1: A single integer that is the minimum distance the cows must walk to a pasture with a sugar cube.

SAMPLE OUTPUT (file butter.out)

```
8
```

OUTPUT DETAILS:

Putting the cube in pasture 4 means: cow 1 walks 3 units; cow 2 walks 5

units; cow 3 walks 0 units -- a total of 8.

Section 3.3

3.3.1 Riding the Fences

Farmer John owns a large number of fences that must be repaired annually. He traverses the fences by riding a horse along each and every one of them (and nowhere else) and fixing the broken parts.

Farmer John is as lazy as the next farmer and hates to ride the same fence twice. Your program must read in a description of a network of fences and tell Farmer John a path to traverse each fence length exactly once, if possible. Farmer J can, if he wishes, start and finish at any fence intersection.

Every fence connects two fence intersections, which are numbered inclusively from 1 through 500 (though some farms have far fewer than 500 intersections). Any number of fences (>=1) can meet at a fence intersection. It is always possible to ride from any fence to any other fence (i.e., all fences are "connected").

Your program must output the path of intersections that, if interpreted as a base 500 number, would have the smallest magnitude.

There will always be at least one solution for each set of input data supplied to your program for testing.

PROGRAM NAME: fence

INPUT FORMAT

Line 1: The number of fences, F (1 <= F <= 1024)

Line 2..F+1: A pair of integers (1 <= i,j <= 500) that tell which pair of intersections this fence connects.

SAMPLE INPUT (file fence.in)

```
9
1 2
2 3
3 4
4 2
4 5
2 5
5 6
5 7
4 6
```

OUTPUT FORMAT

The output consists of F+1 lines, each containing a single integer. Print the number of the starting intersection on the first

line, the next intersection's number on the next line, and so on, until the final intersection on the last line. There might be many possible answers to any given input set, but only one is ordered correctly.

SAMPLE OUTPUT (file fence.out)

```
1
2
3
4
2
5
4
6
5
7
```

3.3.2 Shopping Offers

IOI'95

In a certain shop, each kind of product has an integer price. For example, the price of a flower is 2 zorkmids (z) and the price of a vase is 5z. In order to attract more customers, the shop introduces some special offers.

A special offer consists of one or more product items together for a reduced price, also an integer. Examples:

- three flowers for 5z instead of 6z, or
- two vases together with one flower for 10z instead of 12z.

Write a program that calculates the price a customer has to pay for a purchase, making optimal use of the special offers to make the price as low as possible. You are not allowed to add items, even if that would lower the price.

For the prices and offers given above, the (lowest) price for three flowers and two vases is 14z: two vases and one flower for the reduced price of 10z and two flowers for the regular price of 4z.

PROGRAM NAME: shopping

INPUT FORMAT

The input file has a set of offers followed by a purchase.

Line 1:	s, the number of special offers, (0 <= s <= 99).
Line 2..s+1:	Each line describes an offer using several integers. The first integer is n (1 <= n <= 5), the number of products that are offered. The subsequent n pairs of integers c and k indicate that k items (1 <= k <= 5) with product code c (1 <= c <= 999) are part of the offer. The last number p on the line stands for the reduced price (1 <= p <= 9999). The reduced price of an offer is less than the sum of the regular prices.
Line s+2:	The first line contains the number b (0 <= b <= 5) of different kinds of products to be purchased.
Line s+3..s+b+2:	Each of the subsequent b lines contains three values: c, k, and p. The value c is the (unique) product code (1 <= c <= 999). The value k indicates how many items of this product are to be purchased (1 <= k <= 5). The value p is the regular price per item (1 <= p <= 999). At most 5*5=25 items can be in the basket.

SAMPLE INPUT (file shopping.in)

```
2
1 7 3 5
2 7 1 8 2 10
2
7 3 2
8 2 5
```

OUTPUT FORMAT

A single line with one integer: the lowest possible price to be paid for the purchases.

SAMPLE OUTPUT (file shopping.out)

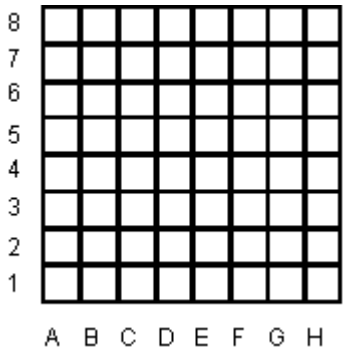
```
14
```

3.3.3 Camelot

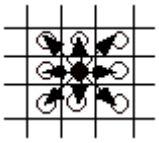
IOI 98

Centuries ago, King Arthur and the Knights of the Round Table used to meet every year on New Year's Day to celebrate their fellowship. In remembrance of these events, we consider a board game for one player, on which one chesspiece king and several knight pieces are placed on squares, no two knights on the same square.

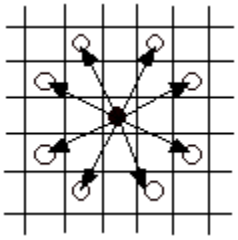
This example board is the standard 8x8 array of squares:



The King can move to any adjacent square from ● to ○ as long as it does not fall off the board:



A Knight can jump from ● to ○, as long as it does not fall off the board:



During the play, the player can place more than one piece in the same square. The board squares are assumed big enough so that a piece is never an obstacle for any other piece to move freely.

The player's goal is to move the pieces so as to gather them all in the same square - in the minimal number of moves. To achieve this, he must move the pieces as prescribed above. Additionally, whenever the king and one or more knights are placed in the same square, the player may choose to move the king and one of the knights together from that point on, as a single knight, up to the final gathering point. Moving the knight together with the king counts as a single move.

Write a program to compute the minimum number of moves the player must perform to produce the gathering. The pieces can gather on any square, of course.

PROGRAM NAME: camelot

INPUT FORMAT

Line 1: Two space-separated integers: R,C, the number of rows and columns on the board. There will be no more than 26 columns and no more than 30 rows.

Line 2..end: The input file contains a sequence of space-separated letter/digit pairs, 1 or more per line. The first pair represents the board position of the king; subsequent pairs represent positions of knights. There might be 0

knights or the knights might fill the board. Rows are numbered starting at 1; columns are specified as upper case characters starting with `A'.

SAMPLE INPUT (file camelot.in)

```
8 8
D 4
A 3 A 8
H 1 H 8
```

The king is positioned at D4. There are four knights, positioned at A3, A8, H1, and H8.

OUTPUT FORMAT

A single line with the number of moves to aggregate the pieces.

SAMPLE OUTPUT (file camelot.out)

```
10
```

SAMPLE OUTPUT ELABORATION

They gather at B5.

Knight 1: A3 - B5 (1 move)

Knight 2: A8 - C7 - B5 (2 moves)

Knight 3: H1 - G3 - F5 - D4 (picking up king) - B5 (4 moves)

Knight 4: H8 - F7 - D6 - B5 (3 moves)

1 + 2 + 4 + 3 = 10 moves.

3.3.4 Home on the Range

Farmer John grazes his cows on a large, square field N (2 <= N <= 250) miles on a side (because, for some reason, his cows will only graze on precisely square land segments). Regrettably, the cows have ravaged some of the land (always in 1 mile square increments). FJ needs to map the remaining squares (at least 2x2 on a side) on which his cows can graze (in these larger squares, no 1x1 mile segments are ravaged).

Your task is to count up all the various square grazing areas within the supplied dataset and report the number of square grazing areas (of sizes >= 2x2) remaining. Of course, grazing areas may overlap for purposes of this report.

PROGRAM NAME: range

INPUT FORMAT

Line 1: N, the number of miles on each side of the field.

Line 2..N+1: N characters with no spaces. 0 represents "ravaged for that block; 1 represents "ready to eat".

SAMPLE INPUT (file range.in)

```
6
101111
001111
111111
001111
101101
111001
```

OUTPUT FORMAT

Potentially several lines with the size of the square and the number of such squares that exist. Order them in ascending order from smallest to largest size.

SAMPLE OUTPUT (file range.out)

```
2 10
3 4
4 1
```

3.3.5 A Game

IOI'96 - Day 1

Consider the following two-player game played with a sequence of N positive integers (2 <= N <= 100) laid onto a game board. Player 1 starts the game. The players move alternately by selecting a number from either the left or the right end of the sequence. That number is then deleted from the board, and its value is added to the score of the player who selected it. A player wins if his sum is greater than his opponents.

Write a program that implements the optimal strategy. The optimal strategy yields maximum points when playing against the "best possible" opponent. Your program must further implement an optimal strategy for player 2.

PROGRAM NAME: game1

INPUT FORMAT

Line 1: N, the size of the board

Line 2-etc: N integers in the range (1..200) that are the contents of the game board, from left to right

SAMPLE INPUT (file game1.in)

```
6
4 7 2 9
5 2
```

OUTPUT FORMAT

Two space-separated integers on a line: the score of Player 1 followed by the score of Player 2.

SAMPLE OUTPUT (file game1.out)

```
18 11
```

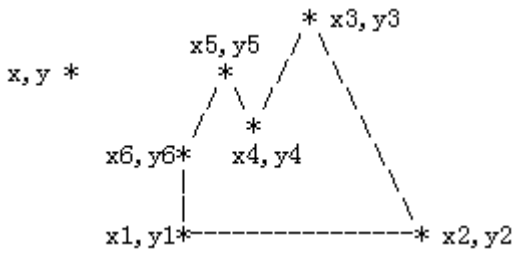
Section 3.4

3.4.1 Closed Fences

A closed fence in the plane is a set of non-crossing, connected line segments with N corners (3 < N < 200). The corners or vertices are each distinct and are listed in counter-clockwise order in an array $\{x_i, y_i\}$, i in (1..N).

Every pair of adjacent vertices defines a side of the fence. Thus $\{x_i\ y_i\ x_{i+1}\ y_{i+1}\}$ is a side of the fence for all i in (1..N). For our purposes, N+1 = 1, so that the first and last vertices making the fence closed.

Here is a typical closed fence and a point x,y:



Write a program which will do the following:

- Test an ordered list of vertices $\{x_i,y_i\}$, i in (1..N) to see if the array is a valid fence.
- Find the set of fence sides that a person (with no height) who is standing in the plane at position (x,y) can "see" when looking at the fence. The location x,y may fall anywhere not on the fence.

A fence side can be seen if there exists a ray that connects (x,y) and any point on the side, and the ray does not intersect any other side of the fence. A side that is parallel to the line of sight is not considered visible. In the figure, above the segments x3,y3-x4,y4; x5,y5-x6,y6; and x6-y6-x1,y1 are visible or partially visible from x,y.

PROGRAM NAME: fence4

INPUT FORMAT

Line 1: N, the number of corners in the fence

Line 2: Two space-separated integers, x and y, that are the location of the observer. Both integers will fit into 16 bits.

Line 3-N+2: A pair of space-separated integers denoting the X,Y location of the corner. The pairs are given in counterclockwise order. Both integers are no larger than 1000 in magnitude.

NOTE: I have added an New test case #12 for this task. Let me know if you think it's wrong. Rob Be sure to include USACO in your mail subject!

SAMPLE INPUT (file fence4.in)

```
13
5 5
0 0
7 0
5 2
7 5
5 7
3 5
4 9
1 8
2 5
0 9
-2 7
0 3
-3 1
```

OUTPUT FORMAT

If the sequence is not a valid fence, the output is a single line containing the word "NOFENCE".

Otherwise, the output is a listing of visible fence segments, one per line, shown as four space-separated integers that represent the two corners. Express the points in the segment by showing first the point that is earlier in the input, then the point that is later. Sort the segments for output by examining the last point and showing first those points that are earlier in the input. Use the same rule on the first of the two points in case of ties.

SAMPLE OUTPUT (file fence4.out)

```
7
0 0 7 0
5 2 7 5
7 5 5 7
5 7 3 5
-2 7 0 3
0 0 -3 1
0 3 -3 1
```

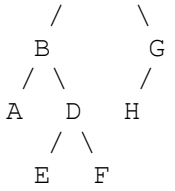
3.4.2 American Heritage

Farmer John takes the heritage of his cows very seriously. He is not, however, a truly fine bookkeeper. He keeps his cow genealogies as binary trees and, instead of writing them in graphic form, he records them in the more linear `tree in-order' and `tree pre-order' notations.

Your job is to create the `tree post-order' notation of a cow's heritage after being given the in-order and pre-order notations. Each cow name is encoded as a unique letter. (You may already know that you can frequently reconstruct a tree from any two of the ordered traversals.) Obviously, the trees will have no more than 26 nodes.

Here is a graphical representation of the tree used in the sample input and output:

```
      C
    /   \
```



The in-order traversal of this tree prints the left sub-tree, the root, and the right sub-tree.

The pre-order traversal of this tree prints the root, the left sub-tree, and the right sub-tree.

The post-order traversal of this tree print the left sub-tree, the right sub-tree, and the root.

PROGRAM NAME: heritage

INPUT FORMAT

Line 1: The in-order representation of a tree.

Line 2: The pre-order representation of that same tree.

SAMPLE INPUT (file heritage.in)

```
ABEDFCHG
CBADEFGH
```

OUTPUT FORMAT

A single line with the post-order representation of the tree.

SAMPLE OUTPUT (file heritage.out)

```
AEFDBHGC
```

3.4.3 Electric Fences

Kolstad & Schrijvers

Farmer John has decided to construct electric fences. He has fenced his fields into a number of bizarre shapes and now must find the optimal place to locate the electrical supply to each of the fences.

A single wire must run from some point on each and every fence to the source of electricity. Wires can run through other fences or across other wires. Wires can run at any angle. Wires can run from any point on a fence (i.e., the ends or anywhere in between) to the electrical supply.

Given the locations of all F (1 <= F <= 150) fences (fences are always parallel to a grid axis and run from one integer gridpoint to another, 0 <= X,Y <= 100), your program must calculate both the total length of wire required to connect every fence to the central source of electricity and also the optimal location for the electrical source.

The optimal location for the electrical source might be anywhere in Farmer John's field, not necessarily on a grid point.

PROGRAM NAME: fence3

INPUT FORMAT

The first line contains F, the number of fences.
F subsequent lines each contain two X,Y pairs each of which denotes the endpoints of a fence.

SAMPLE INPUT (file fence3.in)

```
3
0 0 0 1
2 0 2 1
0 3 2 3
```

OUTPUT FORMAT

On a single line, print three space-separated floating point numbers, each with a single decimal place. Presume that your computer's output library will round the number correctly.

The three numbers are:

- the X value of the optimal location for the electricity,
- the Y value for the optimal location for the electricity, and
- the total (minimum) length of the wire required.

SAMPLE OUTPUT (file fence3.out)

```
1.0 1.6 3.7
```

3.4.4 Raucous Rockers

You just inherited the rights to N (1 <= N <= 20) previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of M (1 <= M <= 20) compact disks with a selection of these songs. Each disk can hold a maximum of T (1 <= T <= 20) minutes of music, and a song can not overlap from one disk to another.

Since you are a classical music fan and have no way to judge the artistic merits of these songs, you decide on the following criteria for making the selection:

- The songs on the set of disks must appear in the order of the dates that they were written.
- The total number of songs included will be maximized.

PROGRAM NAME: rockers

INPUT FORMAT

Line 1: Three integers: N, T, and M.

Line 2: N integers that are the lengths of the songs ordered by the date they were written.

SAMPLE INPUT (file rockers.in)

```
4 5 2
4 3 4 2
```

OUTPUT FORMAT

A single line with an integer that is the number of songs that will fit on M disks.

SAMPLE OUTPUT (file rockers.out)

```
3
```

Chapter 4 Advanced algorithms and difficult drills

Section 4.1

4.1.1 Beef McNuggets

Hubert Chen

Farmer Brown's cows are up in arms, having heard that McDonalds is considering the introduction of a new product: Beef McNuggets. The cows are trying to find any possible way to put such a product in a negative light.

One strategy the cows are pursuing is that of `inferior packaging'. ``Look,'' say the cows, ``if you have Beef McNuggets in boxes of 3, 6, and 10, you can not satisfy a customer who wants 1, 2, 4, 5, 7, 8, 11, 14, or 17 McNuggets. Bad packaging: bad product.''

Help the cows. Given N (the number of packaging options, 1 <= N <= 10), and a set of N positive integers (1 <= i <= 256) that represent the number of nuggets in the various packages, output the largest number of nuggets that can not be purchased by buying nuggets in the given sizes. Print 0 if all possible purchases can be made or if there is no bound to the largest number.

The largest impossible number (if it exists) will be no larger than 2,000,000,000.

PROGRAM NAME: nuggets

INPUT FORMAT

Line 1: N, the number of packaging options

Line 2..N+1: The number of nuggets in one kind of box

SAMPLE INPUT (file nuggets.in)

```
3
3
6
10
```

OUTPUT FORMAT

The output file should contain a single line containing a single integer that represents the largest number of nuggets that can not be represented or 0 if all possible purchases can be made or if there is no bound to the largest number.

SAMPLE OUTPUT (file nuggets.out)

```
17
```

4.1.2 Fence Rails

Burch, Kolstad, and Schrijvers

Farmer John is trying to erect a fence around part of his field. He has decided on the shape of the fence and has even already installed the posts, but he's having a problem with the rails. The local lumber store has dropped off boards of varying lengths; Farmer John must create as many of the rails he needs from the supplied boards.

Of course, Farmer John can cut the boards, so a 9 foot board can be cut into a 5 foot rail and a 4 foot rail (or three 3 foot rails, etc.). Farmer John has an `ideal saw', so ignore the `kerf' (distance lost during sawing); presume that perfect cuts can be made.

The lengths required for the rails might or might not include duplicates (e.g., a three foot rail and also another three foot rail might both be required). There is no need to manufacture more rails (or more of any kind of rail) than called for the list of required rails.

PROGRAM NAME: fence8

INPUT FORMAT

Line 1:	N (1 <= N <= 50), the number of boards
Line 2..N+1:	N lines, each containing a single integer that represents the length of one supplied board
Line N+2:	R (1 <= R <= 1023), the number of rails
Line N+3..N+R+1:	R lines, each containing a single integer (1 <= r_i <= 128) that represents the length of a single required fence rail

SAMPLE INPUT (file fence8.in)

```
4
30
40
50
25
10
15
16
17
18
19
20
21
25
24
30
```

OUTPUT FORMAT

A single integer on a line that is the total number of fence rails that can be cut from the supplied boards. Of course, it might not be possible to cut all the possible rails from the given boards.

SAMPLE OUTPUT (file fence8.out)

```
7
```

4.1.3 Fence Loops

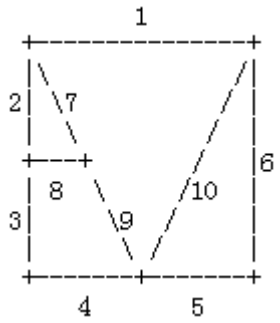
The fences that surround Farmer Brown's collection of pastures have gotten out of control. They are made up of straight segments from 1 through 200 feet long that join together only at their endpoints though sometimes more than two fences join together at a given endpoint. The result is a web of fences enclosing his pastures. Farmer Brown wants to start to straighten things out. In particular, he wants to know which of the pastures has the smallest perimeter.

Farmer Brown has numbered his fence segments from 1 to N (N = the total number of segments). He knows the following about each fence segment:

- the length of the segment
- the segments which connect to it at one end
- the segments which connect to it at the other end.

Happily, no fence connects to itself.

Given a list of fence segments that represents a set of surrounded pastures, write a program to compute the smallest perimeter of any pasture. As an example, consider a pasture arrangement, with fences numbered 1 to 10 that looks like this one (the numbers are fence ID numbers):



The pasture with the smallest perimeter is the one that is enclosed by fence segments 2, 7, and 8.

PROGRAM NAME: fence6

INPUT FORMAT

Line 1: N (1 <= N <= 100)

Line 2..3*N+1: N sets of three line records:

The first line of each record contains four integers: s, the segment number (1 <= s <= N); L_s, the length of the segment (1 <= L_s <= 255); $N1_s$ (1 <= $N1_s$ <= 8) the number of items on the subsequent line; and $N2_s$the number of items on the line after that (1 <= $N2_s$ <= 8).

The second line of the record contains N1 integers, each representing a connected line segment on one end of the fence.

The third line of the record contains N2 integers, each representing a connected line segment on the other end of the fence.

SAMPLE INPUT (file fence6.in)

```
10
1 16 2 2
2 7
10 6
2 3 2 2
1 7
8 3
3 3 2 1
8 2
4
4 8 1 3
3
9 10 5
5 8 3 1
9 10 4
6
6 6 1 2
5
1 10
7 5 2 2
1 2
8 9
8 4 2 2
2 3
7 9
9 5 2 3
7 8
4 5 10
10 10 2 3
1 6
4 9 5
```

OUTPUT FORMAT

The output file should contain a single line with a single integer that represents the shortest surrounded perimeter.

SAMPLE OUTPUT (file fence6.out)

```
12
```

4.1.4 Cryptcowgraphy

Brian Dean

The cows of Farmer Brown and Farmer John are planning a coordinated escape from their respective farms and have devised a method of encryption to protect their written communications.

Specifically, if one cow has a message, say, "International Olympiad in Informatics", it is altered by inserting the letters C, O, and W, in random location in the message, such that C appears before O, which appears before W. Then the cows take the part of the message between C and O, and the part between O and W, and swap them. Here are two examples:

```
International Olympiad in Informatics
                  ->
CnOIWternational Olympiad in Informatics

International Olympiad in Informatics
                  ->
International Cin InformaticsOOlympiad W
```

To make matters more difficult, the cows can apply their encryption scheme several times, by again encrypting the string that results from the previous encryption. One night, Farmer John's cows receive such a multiply-encrypted message. Write a program to compute whether or not the non-encrypted original message could have been the string:

```
Begin the Escape execution at the Break of Dawn
```

PROGRAM NAME: cryptcow

INPUT FORMAT

A single line (with both upper and lower case) with no more than 75 characters that represents the encrypted message.

SAMPLE INPUT (file cryptcow.in)

```
Begin the EscCution at the BreOape execWak of Dawn
```

OUTPUT FORMAT

Two integers on a single line. The first integer is 1 if the message decodes as an escape message; 0 otherwise. The second integer specifies the number of encryptions that were applied (or 0 if the first integer was 0).

SAMPLE OUTPUT (file cryptcow.out)

```
1 1
```

Section 4.2

4.2.1 Drainage Ditches

Hal Burch

Every time it rains on Farmer John's fields, a pond forms over Bessie's favorite clover patch. This means that the clover is covered by water for awhile and takes quite a long time to regrow. Thus, Farmer John has built a set of drainage ditches so that Bessie's clover patch is never covered in water. Instead, the water is drained to a nearby stream. Being an ace engineer, Farmer John has also installed regulators at the beginning of each ditch, so he can control at what rate water flows into that ditch.

Farmer John knows not only how many gallons of water each ditch can transport per minute but also the exact layout of the ditches, which feed out of the pond and into each other and stream in a potentially complex network. Note however, that

there can be more than one ditch between two intersections.

Given all this information, determine the maximum rate at which water can be transported out of the pond and into the stream. For any given ditch, water flows in only one direction, but there might be a way that water can flow in a circle.

PROGRAM NAME: ditch

INPUT FORMAT

Line 1: Two space-separated integers, N (0 <= N <= 200) and M (2 <= M <= 200). N is the number of ditches that Farmer John has dug. M is the number of intersections points for those ditches. Intersection 1 is the pond. Intersection point M is the stream.

Line 2..N+1: Each of N lines contains three integers, S_i, E_i, and C_i. S_i and E_i (1 <= S_i, E_i <= M) designate the intersections between which this ditch flows. Water will flow through this ditch from S_i to E_i. C_i (0 <= C_i <= 10,000,000) is the maximum rate at which water will flow through the ditch.

SAMPLE INPUT (file ditch.in)

```
5 4
1 2 40
1 4 20
2 4 20
2 3 30
3 4 10
```

OUTPUT FORMAT

One line with a single integer, the maximum rate at which water may emptied from the pond.

SAMPLE OUTPUT (file ditch.out)

```
50
```

4.2.2 The Perfect Stall

Hal Burch

Farmer John completed his new barn just last week, complete with all the latest milking technology. Unfortunately, due to engineering problems, all the stalls in the new barn are different. For the first week, Farmer John randomly assigned cows to stalls, but it quickly became clear that any given cow was only willing to produce milk in certain stalls. For the last week, Farmer John has been collecting data on which cows are willing to produce milk in which stalls. A stall may be only assigned to one cow, and, of course, a cow may be only assigned to one stall.

Given the preferences of the cows, compute the maximum number of milk-producing assignments of cows to stalls that is possible.

PROGRAM NAME: stall4

INPUT FORMAT

Line 1: 3e line with two integers, N (0 <= N <= 200) and M (0 <= M <= 200). N is the number of cows that Farmer John has and M is the number of stalls in the new barn.

Line 2..N+1: N lines, each corresponding to a single cow. The first integer (S_i) on the line is the number of stalls that the cow is willing to produce milk in (0 <= S_i <= M). The subsequent S_i integers on that line are the stalls in which that cow is willing to produce milk. The stall numbers will be integers in the range (1..M), and no stall will be listed twice for a given cow.

SAMPLE INPUT (file stall4.in)

```
5 5
2 2 5
3 2 3 4
2 1 5
3 1 2 5
```

```
1 2
```

OUTPUT FORMAT

A single line with a single integer, the maximum number of milk-producing stall assignments that can be made.

SAMPLE OUTPUT (file stall4.out)

```
4
```

4.2.3 Job Processing

IOI'96

A factory is running a production line that requires two operations to be performed on each job: first operation "A" then operation "B". Only a certain number of machines are capable of performing each operation.

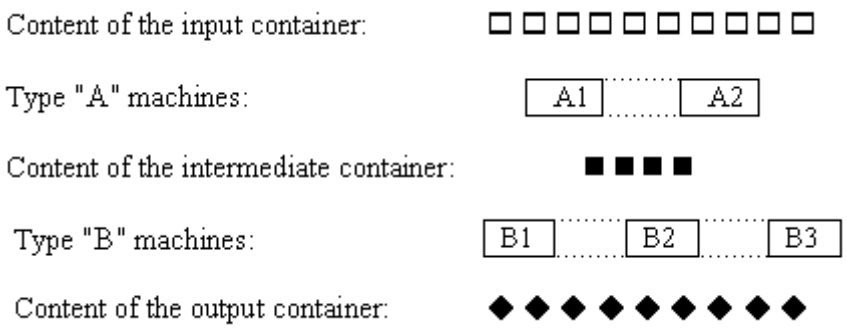


Figure 1 shows the organization of the production line that works as follows. A type "A" machine takes a job from the input container, performs operation "A" and puts the job into the intermediate container. A type "B" machine takes a job from the intermediate container, performs operation "B" and puts the job into the output container. All machines can work in parallel and independently of each other, and the size of each container is unlimited. The machines have different performance characteristics, a given machine requires a given processing time for its operation.

Give the earliest time operation "A" can be completed for all N jobs provided that the jobs are available at time 0. Compute the minimal amount of time that is necessary to perform both operations (successively, of course) on all N jobs.

PROGRAM NAME: job

INPUT FORMAT

Line 1: Three space-separated integers:

- , the number of jobs (1<=N<=1000).
- M1, the number of type "A" machines (1<=M1<=30)
- M2, the number of type "B" machines (1<=M2<=30)

Line 2..etc: M1 integers that are the job processing times of each type "A" machine (1..20) followed by M2 integers, the job processing times of each type "B" machine (1..20).

SAMPLE INPUT (file job.in)

```
5 2 3
1 1 3 1 4
```

OUTPUT FORMAT

A single line containing two integers: the minimum time to perform all "A" tasks and the minimum time to perform all "B" tasks (which require "A" tasks, of course).

SAMPLE OUTPUT (file job.out)

```
3 5
```

4.2.4 Cowcycles

Originally by Don Gillies

[International readers should note that some words are puns on cows.]

Having made a fortune on Playbov magazine, Hugh Heifer has moved from his original field in the country to a fashionable yard in the suburbs. To visit fond pastoral memories, he wishes to cowmmute back to his old stomping grounds. Being environmentally minded, Hugh wishes to transport himself using his own power on a Cowcycle (a bicycle specially fitted for his neatly manicured hooves).

Hugh weighs over a ton; as such, getting smoothly up to speed on traditional cowcycle gear sets is a bit challenging. Changing among some of the widely spaced gear ratios causes exertion that's hard on Hugh's heart.

Help Hugh outfit his Cowcycle by choosing F (1 <= F <= 5) gears (sprockets) in the front and R (1 <= R <= 10) gears in the rear of his F*R speed cowcycle subject to these rules:

- The possible sizes (number of teeth) for the F front gears are specified.
- The possible sizes (number of teeth) for the R rear gears are specified.
- At any given gear setting, the gear ratio is the quotient of the number of teeth on the front gear and the number of teeth on the rear gear (i.e., number of front gear teeth divided by number of rear gear teeth)
- The largest gear ratio must be at least three times the smallest.
- The variance (see below) of the set of DIFFERENCES between successive (i.e., after sorting) gear ratios should be minimized.

Calculate the mean and variance of a set of differences (x_i in this formula) by the following formulae:

$$\text{mean} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$\text{variance} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \text{mean})^2$$

Deduce and print the optimal sets of F front gears and R rear gears so that the variance is minimized (and the ratios span a factor of at least 3x).

PROGRAM NAME: cowcycle

INPUT FORMAT

The first line contains F and R, the numbers of front and rear gears. The second line contains four numbers: F1, F2 (25 <= F1 < F2 <= 80), R1, and R2 (5 <= R1 < R2 <= 40). All front gears from F1 through F2 are available; all rear gears from R1 through R2 are available. There will exist at least one legal set of gears.

SAMPLE INPUT (file cowcycle.in)

```
2 5
39 62 12 28
```

OUTPUT FORMAT

Display the number of teeth on the set of F chosen front gears, from smallest to largest, on the first line of output (separated by spaces). Display the number of teeth on the set of R chosen rear gears, from smallest to largest, on the second line of output. All gears have an integer number of teeth, of course.

If multiple optimal answers exist, output the answer with the smallest front gear set (smallest first gear, or smallest second gear if first gears match, etc.). Likewise, if all first gears match, output the answer with the smallest rear gear set (similar rules to the front gear set).

SAMPLE OUTPUT (file cowcycle.out)

```
39 53
12 13 15 23 27
```

Comment

The challenge in this problem is "reading the problem". Don't read further if you are working on that level of challenge. If the problem is just completely unclear to you, read in.

The problem wants you to find "an optimal set of gear ratios" such that the spacing between the ratios is most uniform. Consider the test case above:

```
2 5
39 62 12 28
```

This specifies two front gears from the set 39..62; five rear gears from the set 12..28. The program must examine all possible pairs of 62-39+1=24 front gears and all possible quintuples from 28-12+1=17 rear gears. Combinatorically, The total number of possibilities is (24 take 2) times (17 take 5), which is 24!/22!/2! x 17!/5!/12! which is 656,880 possibilities (I think).

For each of these possibilities, calculations like the following. This example considers in some sense the "first" case: front gears of 39 and 40, rear gears of 12, 13, 14, 15, and 16.

First, calculate all the possible ratios:

```
39/12 = 3.2500000000000000000
39/13 = 3.0000000000000000000
39/14 = 2.7857142857142857142
39/15 = 2.6000000000000000000
39/16 = 2.4375000000000000000
40/12 = 3.3333333333333333333
40/13 = 3.0769230769230769230
40/14 = 2.8571428571428571428
40/15 = 2.6666666666666666666
40/16 = 2.5000000000000000000
```

Then, sort them:

```
39/16 = 2.4375000000000000000
40/16 = 2.5000000000000000000
39/15 = 2.6000000000000000000
40/15 = 2.6666666666666666666
39/14 = 2.7857142857142857142
40/14 = 2.8571428571428571428
39/13 = 3.0000000000000000000
40/13 = 3.0769230769230769230
39/12 = 3.2500000000000000000
40/12 = 3.3333333333333333333
```

Then, calculate the absolute value of the differences:

```
2.4375000000000000000 - 2.5000000000000000000 = 0.0625000000000000000
2.5000000000000000000 - 2.6000000000000000000 = 0.1000000000000000000
2.6000000000000000000 - 2.6666666666666666666 = 0.0666666666666666666
2.6666666666666666666 - 2.7857142857142857142 = 0.1190476190476190476
2.7857142857142857142 - 2.8571428571428571428 = 0.0714285714285714285
2.8571428571428571428 - 3.0000000000000000000 = 0.1428571428571428571
3.0000000000000000000 - 3.0769230769230769230 = 0.0769230769230769230
3.0769230769230769230 - 3.2500000000000000000 = 0.1730769230769230769
3.2500000000000000000 - 3.3333333333333333333 = 0.0833333333333333333
```

Then, calculate the mean and variance of the set of numbers on the right, above. The mean is (I think): 0.09953703703703703703666666. The variance is approximately 0.00129798488416722.

Of course this set of gears is not valid, since it does not have a 3x span from highest gear to lowest.

Find the set of gears that minimizes the variance and has a 3x or greater span.

Section 4.3

4.3.1 Buy Low, Buy Lower

The advice to "buy low" is half the formula to success in the stock market. But to be considered a great investor you must also follow this problems' advice:

"Buy low, buy lower"

That is, each time you buy a stock, you must purchase more at a lower price than the previous time you bought it. The more times you buy at a lower price than before, the better! Your goal is to see how many times you can continue purchasing at ever lower prices.

You will be given the daily selling prices of a stock over a period of time. You can choose to buy stock on any of the days. Each time you choose to buy, the price must be lower than the previous time you bought stock. Write a program which identifies which days you should buy stock in order to maximize the number of times you buy.

By way of example, suppose on successive days stock is selling like this:

```
 Day   1  2  3  4  5  6  7  8  9 10 11 12
Price 68 69 54 64 68 64 70 67 78 62 98 87
```

In the example above, the best investor (by this problem, anyway) can buy at most four times if they purchase at a lower price each time. One four day sequence (there might be others) of acceptable buys is:

```
Day    2  5  6 10
Price 69 68 64 62
```

PROGRAM NAME: buylow

INPUT FORMAT

Line 1: N (1 <= N <= 5000), the number of days for which stock prices are available.

Line 2..etc: A series of N positive space-separated integers (which may require more than one line of data) that tell the price for that day. The integers will fit into 32 bits quite nicely.

SAMPLE INPUT (file buylow.in)

```
12
68 69 54 64 68 64 70 67
78 62 98 87
```

OUTPUT FORMAT

Two integers on a single line:

- the length of the longest sequence of decreasing prices
- the number of sequences that have this length

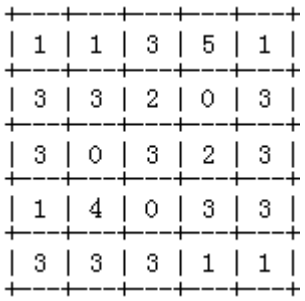
In counting the number of solutions, two potential solutions are considered the same (and would only count as one solution) if they repeat the same string of decreasing prices, that is, if they "look the same" when the successive prices are compared. Thus, two different sequence of "buy" days could produce the same string of decreasing prices and be counted as only a single solution.

SAMPLE OUTPUT (file buylow.out)

```
4 2
```

4.3.2 The Primes

In the square below, each row, each column and the two diagonals can be read as a five digit prime number. The rows are read from left to right. The columns are read from top to bottom. Both diagonals are read from left to right.



- The prime numbers' digits must sum to the same number.
- The digit in the top left-hand corner of the square is pre-determined (1 in the example).
- A prime number may be used more than once in the same square.
- If there are several solutions, all must be presented (sorted in numerical order as if the 25 digits were all one long number).
- A five digit prime number cannot begin with a zero (e.g., 00003 is NOT a five digit prime number).

PROGRAM NAME: prime3

INPUT FORMAT

A single line with two space-separated integers: the sum of the digits and the digit in the upper left hand corner of the square.

SAMPLE INPUT (file prime3.in)

```
11 1
```

OUTPUT FORMAT

Five lines of five characters each for each solution found, where each line in turn consists of a five digit prime number. Print a blank line between solutions. If there are no prime squares for the input data, output a single line containing "NONE".

SAMPLE OUTPUT (file prime3.out)

The above example has 3 solutions.

```
11351
14033
30323
53201
13313

11351
33203
30323
14033
33311

13313
13043
32303
50231
13331
```

4.3.3 Street Race

IOI'95

Figure 1 gives an example of a course for a street race. You see some points, labeled from 0 to N (here, N=9), and some arrows connecting them. Point 0 is the start of the race; point N is the finish. The arrows represent one-way streets. The participants of the race move from point to point via the streets, in the direction of the arrows only. At each point, a participant may choose any outgoing arrow.

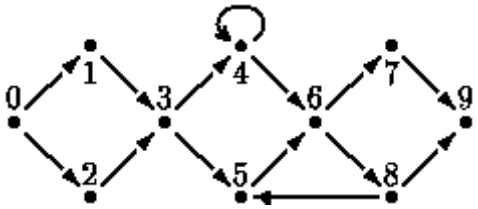


Figure 1: A street course with 10 points

A well-formed course has the following properties:

- Every point in the course can be reached from the start.
- The finish can be reached from each point in the course.
- The finish has no outgoing arrows.

A participant does not have to visit every point of the course to reach the finish. Some points, however, are unavoidable. In the example, these are points 0, 3, 6, and 9. Given a well-formed course, your program must determine the set of unavoidable points that all participants have to visit, excluding start and finish.

Suppose the race has to be held on two consecutive days. For that purpose the course has to be split into two courses, one for each day. On the first day, the start is at point 0 and the finish at some `splitting point'. On the second day, the start is at this splitting point and the finish is at point N. Given a well-formed course, your program must also determine the set of splitting points. A point S is a splitting point for the well-formed course C if S differs from the star t and the finish of C, and the course can be split into two well-formed courses that (1) have no common arrows and (2) have S as their only common point, with S appearing as the finish of one and the start of the other. In the example, only point 3 is a splitting point.

PROGRAM NAME: race3

INPUT FORMAT

The input file contains a well-formed course with at most 50 points and at most 100 arrows. There are N+2 lines in the file. The first N+1 lines contain the endpoints of the arrows that leave from the points 0 through N respectively. Each of these lines ends with the number -2. The last line contains only the number -1.

SAMPLE INPUT (file race3.in)

```
1 2 -2
3 -2
3 -2
5 4 -2
6 4 -2
6 -2
7 8 -2
9 -2
5 9 -2
-2
-1
```

OUTPUT FORMAT

Your program should write two lines. The first line should contain the number of unavoidable points in the input course, followed by the labels of these points, in ascending order. The second line should contain the number of splitting points of the input course, followed by the labels of all these points, in ascending order.

SAMPLE OUTPUT (file race3.out)

```
2 3 6
1 3
```

4.3.4 Letter Game

IOI 1995

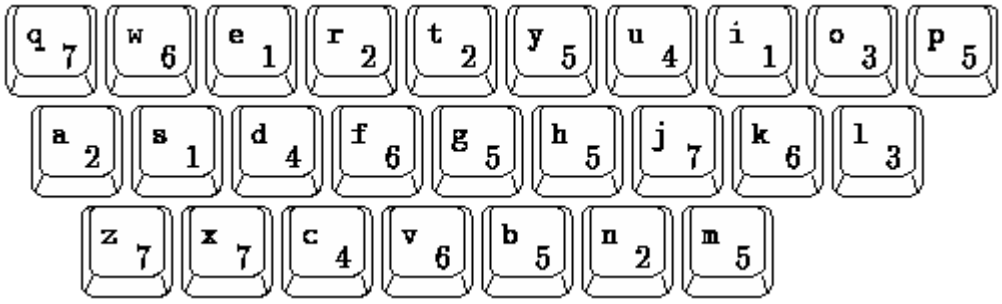


Figure 1: Each of the 26 lowercase letters and its value

Letter games are popular at home and on television. In one version of the game, every letter has a value, and you collect letters to form one or more words giving the highest possible score. Unless you have `a way with words', you will try all the words you know, sometimes looking up the spelling, and then compute the scores. Obviously, this can be done more accurately by computer.

Given the values in Figure 1, a list of words, and the letters collected: find the highest scoring words or pairs of words that can be formed.

PROGRAM NAME: lgame

INPUT FORMAT

One line with a string of lowercase letters (from `a' to `z'). The string consists of at least 3 and at most 7 letters in arbitrary order.

SAMPLE INPUT (file lgame.in)

```
prmgroa
```

DICTIONARY FORMAT

At most 40,000 lines, each containing a string of at least 3 and at most 7 lowercase letters. At the end of this file is a line with a single period (`.'). The file is sorted alphabetically and contains no duplicates.

SAMPLE DICTIONARY (file lgame.dict[ii])

```
profile
program
prom
rag
ram
rom
.
```

OUTPUT FORMAT

On the first line, your program should write the highest possible score, and on each of the following lines, all the words and/or word pairs from file `lgame.dict` with this score. Sort the output alphabetically by first word, and if tied, by second word. A letter must not occur more often in an output line than in the input line. Use the letter values given in Figure 1.

When a combination of two words can be formed with the given letters, the words should be printed on the same line separated by a space. The two words should be in alphabetical order; for example, do not write `rag prom', only write `prom rag'. A pair in an output line may consist of two identical words.

SAMPLE OUTPUT (file lgame.out)

This output uses the tiny dictionary above, not the `lgame.dict` dictionary.

```
24
program
prom rag
```

[ii] See http://ace.delos.com/usaco/lgame.dict.

Section 4.4

4.4.1 Shuttle Puzzle

Traditional

The Shuttle Puzzle of size 3 consists of 3 white marbles, 3 black marbles, and a strip of wood with 7 holes. The marbles of the same color are placed in the holes at the opposite ends of the strip, leaving the center hole empty.

```
INITIAL STATE: WWW_BBB
GOAL STATE: BBB_WWW
```

To solve the shuttle puzzle, use only two types of moves. Move 1 marble 1 space (into the empty hole) or jump 1 marble over 1 marble of the opposite color (into the empty hole). You may not back up, and you may not jump over 2 marbles.

A Shuttle Puzzle of size N consists of N white marbles and N black marbles and 2N+1 holes.

Here's one solution for the problem of size 3 showing the initial, intermediate, and end states:

```
WWW BBB
WW WBBB
WWBW BB
WWBWB B
WWB BWB
W BWBWB
 WBWBWB
BW WBWB
BWBW WB
BWBWBW
BWBWB W
BWB BWW
B BWBWW
BB WBWW
BBBW WW
BBB WWW
```

Write a program that will solve the SHUTTLE PUZZLE for any size N (1 <= N <= 12) in the minimum number of moves and display the successive moves, 20 per line.

PROGRAM NAME: shuttle

INPUT FORMAT

A single line with the integer N.

SAMPLE INPUT (file shuttle.in)

```
3
```

OUTPUT FORMAT

The list of moves expressed as space-separated integers, 20 per line (except possibly the last line). Number the marbles/holes from the left, starting with one.

Output the the solution that would appear first among the set of minimal solutions sorted numerically (first by the first number, using the second number for ties, and so on).

SAMPLE OUTPUT (file shuttle.out)

```
3 5 6 4 2 1 3 5 7 6 4 2 3 5 4
```

4.4.2 Pollutant Control

Hal Burch

It's your first day in Quality Control at Merry Milk Makers, and already there's been a catastrophe: a shipment of bad milk has been sent out. Unfortunately, you didn't discover this until the milk was already into your delivery system on its way to stores. You know which grocer that milk was destined for, but there may be multiple ways for the milk to get to that store.

The delivery system is made up of a several warehouses, with trucks running from warehouse to warehouse moving milk. While the milk will be found quickly, it is important that it does not make it to the grocer, so you must shut down enough trucks to ensure that it is impossible for the milk to get to the grocer in question. Every route costs a certain amount to shut down. Find the minimum amount that must be spent to ensure the milk does not reach its destination, along with a set of trucks to shut down that achieves this goal at that cost.

PROGRAM NAME: milk6

INPUT FORMAT

Line 1: Two space separated integers, N and M. N (2 <= N <= 32) is the number of warehouses that Merry Milk Makers has, and M (0 <= M <= 1000) is the number of trucks routes run. Warehouse 1 is actually the productional facility, while warehouse N is the grocer to which which the bad milk was destined.

Line 2..M+1: Truck routes: three space-separated integers, S_i, E_i, and C_i. S_i and E_i (1 <= S_i,E_i <= N) correspond to the pickup warehouse and dropoff warehouse for the truck route. C_i (0 <= C_i <= 2,000,000) is the cost of shutting down the truck route.

SAMPLE INPUT (file milk6.in)

```
4 5
1 3 100
3 2 50
2 4 60
1 2 40
2 3 80
```

OUTPUT FORMAT

The first line of the output should be two integers, C and T. C is the minimum amount which must be spent in order to ensure the our milk never reaches its destination. T is the minimum number of truck routes that you plan to shut down in order to achive this goal. The next T lines sould contain a sorted list of the indexes of the truck routes that you suggest shutting down. If there are multiple sets of truck routes that achieve the goal at minimum cost, choose one that shuts down the minimum number of routes. If there are still multiple sets, choose the one whose initial routes have the smallest index.

SAMPLE OUTPUT (file milk6.out)

```
60 1
3
```

4.4.3 Frame Up

Consider the following five picture frames shown on an 9 x 8 array:

```
........   ........   ........   ........   .CCC....
EEEEEE..   ........   ........   ..BBBB..   .C.C....
E....E..   DDDDDD..   ........   ..B..B..   .C.C....
E....E..   D....D..   ........   ..B..B..   .CCC....
E....E..   D....D..   ....AAAA   ..B..B..   ........
E....E..   D....D..   ....A..A   ..BBBB..   ........
E....E..   DDDDDD..   ....A..A   ........   ........
E....E..   ........   ....AAAA   ........   ........
EEEEEE..   ........   ........   ........   ........

  1          2          3          4          5
```

Now place all five picture frames on top of one another starting with 1 at the bottom and ending up with 5 on top. If any part of a frame covers another frame, it hides that part of the frame below. Viewing the stack of five frames we see the following.

```
.CCC...
ECBCBB..
DCBCDB..
DCCC.B..
D.B.ABAA
D.BBBB.A
DDDDAD.A
```

```
E...AAAA
EEEEEE..
```

Given a picture like this, determine the order of the frames stacked from bottom to top.

Here are the rules for this challenge:

- The width of the frame is always exactly 1 character and the sides are never shorter than 3 characters.
- It is possible to see at least one part of each of the four sides of a frame. A corner is part of two sides.
- The frames will be lettered with capital letters, and no two frames will be assigned the same letter.

PROGRAM NAME: frameup

INPUT FORMAT

Line 1: Two space-separated integers: the height H (3 <= H <=30) and the width W (3 <= W <= 30).

Line 2..H+1: H lines, each with a string W characters wide.

SAMPLE INPUT (file frameup.in)

```
9 8
.CCC....
ECBCBB..
DCBCDB..
DCCC.B..
D.B.ABAA
D.BBBB.A
DDDDAD.A
E...AAAA
EEEEEE..
```

OUTPUT FORMAT

Print the letters of the frames in the order they were stacked from bottom to top. If there are multiple possibilities for an ordering, list all such possibilities -- in alphabetical order -- on successive lines. There will always be at least one legal ordering.

SAMPLE OUTPUT (file frameup.out)

```
EDABC
```

Chapter 5 Serious Challenges

Section 5.1

5.1.1 Fencing the Cows

Hal Burch

Farmer John wishes to build a fence to contain his cows, but he's a bit short on cash right. Any fence he builds must contain all of the favorite grazing spots for his cows. Given the location of these spots, determine the length of the shortest fence which encloses them.

PROGRAM NAME: fc

INPUT FORMAT

The first line of the input file contains one integer, N. N (0 <= N <= 10,000) is the number of grazing spots that Farmer john wishes to enclose. The next N line consists of two real numbers, Xi and Yi, corresponding to the location of the grazing spots in the plane (-1,000,000 <= Xi,Yi <= 1,000,000). The numbers will be in decimal format.

SAMPLE INPUT (file fc.in)

```
4
4 8
4 12
5 9.3
7 8
```

OUTPUT FORMAT

The output should consists of one real number, the length of fence required. The output should be accurate to two decimal places.

SAMPLE OUTPUT (file fc.out)

```
12.00
```

5.1.2 Starry Night

IOI' 98

High up in the night sky, the shining stars appear in clusters of various shapes. A **cluster** is a non-empty group of neighbouring stars, adjacent in horizontal, vertical or diagonal direction. A cluster cannot be a part of a larger cluster.

Clusters may be similar. Two clusters are **similar** if they have the same shape and number of stars, irrespective of their orientation. In general, the number of possible orientations for a cluster is eight, as Figure 1 exemplifies.

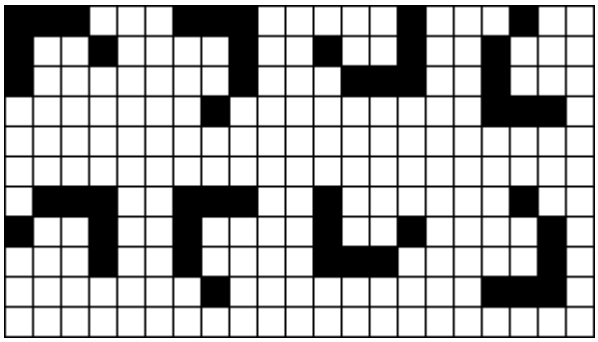


Figure 1. Eight similar clusters

The night sky is represented by a sky map, which is a two-dimensional matrix of 0's and 1's. A cell contains the digit 1 if it has a star, and the digit 0 otherwise.

Given a sky map, mark all the clusters with lower case letters. Similar clusters must be marked with the same letter;

non-similar clusters must be marked with different letters.

You mark a cluster with a lower case letter by replacing every 1 in the cluster by that lower case letter.

PROGRAM NAME: starry

INPUT FORMAT

The first two lines contain, respectively, the width **W** and the height **H** of a sky map. The sky map is given in the following **H** lines, of **W** characters each.

SAMPLE INPUT (file starry.in)

```
23
15
10001000000000010000000
01111100011111000101101
01000000010001000111111
00000000010101000101111
00000111010001000000000
00001001011111000000000
10000001000000000000000
00101000000111110010000
00001000000100010011111
00000001110101010100010
00000100110100010000000
00010001110111110000000
00100001110000000100000
00001000100001000100101
00000001110001000111000
```

In this case, the sky map has width 23 and height 15. Just to make it clearer, notice that this input file corresponds to the following picture of the sky.

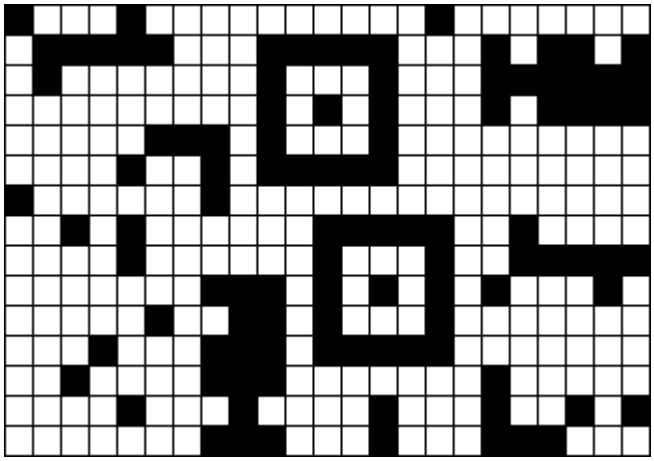


Figure 2. Picture of the sky

OUTPUT FORMAT

The output file contains the same map as the input file, except that the clusters are marked as described in Task.

There will generally be more than one way to label the clusters with letters. Your program should choose the labeling such that if the entire output file is read as a string, this string will be minimal in the lexicographical ordering.

SAMPLE OUTPUT (file starry.out)

```
a000a0000000000b0000000
0aaaaa000ccccc000d0dd0d
0a0000000c000c000dddddd
000000000c0b0c000d0dddd
00000eee0c000c000000000
0000e00e0ccccc000000000
b000000e000000000000000
00b0f000000ccccc00a0000
0000f000000c000c00aaaaa
0000000ddd0c0b0c0a000a0
```

```
00000b00dd0c000c0000000
000g000ddd0ccccc0000000
00g0000ddd0000000e00000
0000b000d0000f000e00e0b
0000000ddd000f000eee000
```

This is one possible result for the sample input above. Notice that this output file corresponds to the following picture.

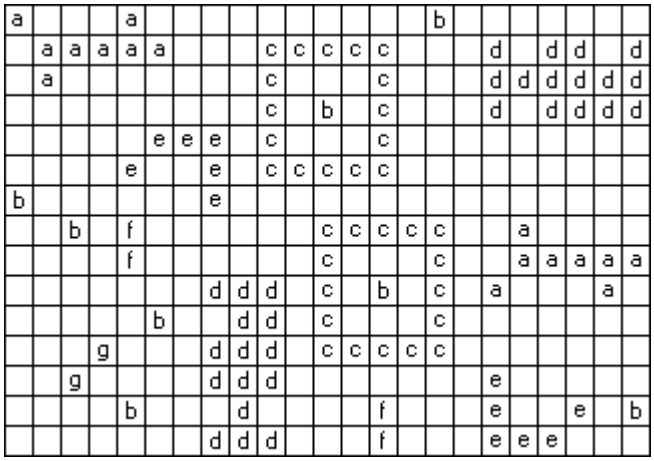


Figure 3. Picture with the clusters marked

Constraints

0 <= **W** (width of the sky map) <= 100

0 <= **H** (height of the sky map) <= 100

0 <= Number of clusters <= 500

0 <= Number of non-similar clusters <= 26 (a..z)

1 <= Number of stars per cluster <= 160

5.1.3 Musical Themes

Brian Dean

A musical melody is represented as a sequence of N (1 <= N <= 5000) notes that are integers in the range 1..88, each representing a key on the piano. It is unfortunate but true that this representation of melodies ignores the notion of musical timing; but, this programming task is about notes and not timings.

Many composers structure their music around a repeating "theme", which, being a subsequence of an entire melody, is a sequence of integers in our representation. A subsequence of a melody is a theme if it:

- is at least five notes long
- appears (potentially transposed -- see below) again somewhere else in the piece of music
- is disjoint from (i.e., non-overlapping with) at least one of its other appearance(s)

Transposed means that a constant positive or negative value is added to every note value in the theme subsequence.

Given a melody, compute the length (number of notes) of the longest theme.

One second time limit for this problem's solutions!

PROGRAM NAME: theme

INPUT FORMAT

The first line of the input file contains the integer N. Each subsequent line (except potentially the last) contains 20 integers representing the sequence of notes. The last line contains the remainder of the notes, potentially fewer than 20.

SAMPLE INPUT (file theme.in)

```
30
25 27 30 34 39 45 52 60 69 79 69 60 52 45 39 34 30 26 22 18
82 78 74 70 66 67 64 60 65 80
```

OUTPUT FORMAT

The output file should contain a single line with a single integer that represents the length of the longest theme. If there are no themes, output 0.

SAMPLE OUTPUT (file theme.out)

```
5
```

[The five-long theme is the last five notes of the first line and the first five notes of the second]

Section 5.2

5.2.1 Snail Trails

All Ireland Contest

Sally Snail likes to stroll on a N x N square grid (1 <n <= 120). She always starts in the upper left corner of the grid. The grid has empty squares (denoted below by `.') and a number (B) of barriers (denoted below by `#'). Here is a depiction of a grid including a demonstration of the grid labelling algorithm:

```
  A B C D E F G H
1 S . . . . . # .
2 . . . . # . . .
3 . . . . . . . .
4 . . . . . . . .
5 . . . . . # . .
6 # . . . . . . .
7 . . . . . . . .
8 . . . . . . . .
```

Sally travels vertically (up or down) or horizontally (left or right). Sally can travel either down or right from her starting location, which is always A1.

Sally travels as long as she can in her chosen direction. She stops and turns 90 degrees whenever she encounters the edge of the board or one of the barriers. She can not leave the grid or enter a space with a barrier. Additionally, Sally can not re-cross any square she has already traversed. She stops her traversal altogether any time she can no longer make a move.

Here is one sample traversal on the sample grid above:

```
  A B C D E F G H
1 S--------+ # .
2 . . . . # | . .
3 . . . . . | . .
4 . . . . . +---+
5 . . . . . # . |
6 # . . . . . . |
7 +-----------+ |
8 +-------------+
```

Sally traversed right, down, right, down, left, up, and right. She could not continue since she encountered a square already visited. Things might have gone differently if she had chosen to turn back toward our left when she encountered the barrier at F5.

Your task is to determine and print the largest possible number of squares that Sally can visit if she chooses her turns wisely. Be sure to count square A1 as one of the visited squares.

PROGRAM NAME: snail

INPUT FORMAT

The first line of the input has N, the dimension of the square, and B, the number of barriers (1 <= B <= 200). The subsequent B lines contain the locations of the barriers. The sample input file below describes the sample grid above. The sample output file below is supposed to describe the traversal shown above. Note that when N > 26 then the input file can not specify barriers to the right of column Z.

SAMPLE INPUT (file snail.in)

```
8 4
```

```
E2
A6
G1
F5
```

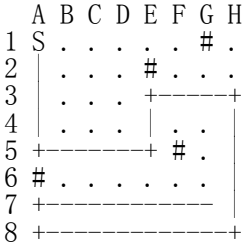
OUTPUT FORMAT

The output file should consist of exactly one line, the largest possible number of squares that Sally can visit.

SAMPLE OUTPUT (file snail.out)

```
33
```

Using this traversal:



5.2.2 Electric Fences

Kolstad & Schrijvers

Farmer John has decided to construct electric fences. He has fenced his fields into a number of bizarre shapes and now must find the optimal place to locate the electrical supply to each of the fences.

A single wire must run from some point on each and every fence to the source of electricity. Wires can run through other fences or across other wires. Wires can run at any angle. Wires can run from any point on a fence (i.e., the ends or anywhere in between) to the electrical supply.

Given the locations of all F (1 <= F <= 150) fences (fences are always parallel to a grid axis and run from one integer gridpoint to another, 0 <= X,Y <= 100), your program must calculate both the total length of wire required to connect every fence to the central source of electricity and also the optimal location for the electrical source.

The optimal location for the electrical source might be anywhere in Farmer John's field, not necessarily on a grid point.

PROGRAM NAME: fence3

INPUT FORMAT

The first line contains F, the number of fences.

F subsequent lines each contain two X,Y pairs each of which denotes the endpoints of a fence.

SAMPLE INPUT (file fence3.in)

```
3
0 0 0 1
2 0 2 1
0 3 2 3
```

OUTPUT FORMAT

On a single line, print three space-separated floating point numbers, each with a single decimal place. Presume that your computer's output library will round the number correctly.

The three numbers are:

- the X value of the optimal location for the electricity,
- the Y value for the optimal location for the electricity, and
- the total (minimum) length of the wire required.

SAMPLE OUTPUT (file fence3.out)

```
1.0 1.6 3.7
```

5.2.3 Wisconsin Squares

It's spring in Wisconsin and time to move the yearling calves to the yearling pasture and last year's yearlings to the greener pastures of the north 40.

Farmer John has five kinds of cows on his farm (abbreviations are shown in parentheses): Guernseys (A), Jerseys (B), Herefords (C), Black Angus (D), and Longhorns (E). These herds are arranged on the 16 acre pasture, one acre for each small herd, on a 4 x 4 grid (labeled with rows and columns) like this:

```
  1 2 3 4
 +-------
1|A B A C
2|D C D E
3|B E B C
4|C A D E
```

In the initial pasture layout, the herds total 3 A's, 3 B's, 4 C's, 3 D's, and 3 E's. This year's calves have one more D herd and one fewer C herd, for a total of 3 A's, 3 B's, 3 C's, 4 D's, and 3 E's.

FJ is extremely careful in his placement of herds onto his pasture grid. This is because when herds of the same types of cows are too close together, they misbehave: they gather near the fence and smoke cigarettes and drink milk. Herds are too close together when they are on the same square or in any of the eight adjacent squares.

Farmer John must move his old herd out of the field and his new herd into the field using his old brown Ford pickup truck, which holds one small herd at a time. He picks up a new herd, drives to a square in the yearling pasture, unloads the new herd, loads up the old herd, and drives the old herd to the north 40 where he unloads it. He repeats this operation 16 times and then drives to Zack's for low-fat yogurt treats and familiar wall decor.

Help Farmer John. He must choose just exactly the correct order to replace the herds so that he never puts a new herd in a square currently occupied by the same type of herd or adjacent to a square occupied by the same type of herd. Of course, once the old cows are gone and the new cows are in place, he must be careful in the future to separate herds based on the new arrangement.

Very important hint: Farmer John knows from past experience that he must move a herd of D cows first.

Find a way for Farmer John to move the yearlings to their new pasture. Print the 16 sequential herd-type/row/column movements that lead to a safe moving experience for the cows.

Calculate the total number of possible final arrangements for the 4x4 pasture and calculate the total number of ways those arrangements can be created.

PROGRAM NAME: wissqu

TIME LIMIT: 5 seconds

INPUT FORMAT

Four lines, each with four letters that denote herds.

SAMPLE INPUT (file wissqu.in)

```
ABAC
DCDE
BEBC
CADE
```

OUTPUT FORMAT

16 lines, each with a herd-type, row and column. If there are multiple solutions (and there are), you should output the solution for which the concatenated string ("D41C42A31 ... D34") of the answers is first in lexicographic order.

One more line with the total number of ways these arrangements can be created.

SAMPLE OUTPUT (file wissqu.out)

```
D 4 1
C 4 2
A 3 1
A 3 3
```

```
B 2 4
B 3 2
B 4 4
E 2 1
E 2 3
D 1 4
D 2 2
C 1 1
C 1 3
A 1 2
E 4 3
D 3 4
14925
```

Section 5.3

5.3.1 Milk Measuring

Hal Burch

Farmer John must measure Q (1 <= Q <= 20,000) quarts of his finest milk and deliver it in one big bottle to a customer. He fills that bottle with exactly the number of quarts that the customer orders.

Farmer John has always been frugal. He is at the cow hardware store where he must purchase a set of pails with which to measure out Q quarts of milk from his giant milk tank. Since the pails each cost the same amount, your task is to figure out a minimal set of pails Farmer John can purchase in order to fill a bottle with exactly Q quarts of milk. Additionally, since Farmer John has to carry the pails home, given two minimal sets of pails he should choose the "smaller" one as follows: Sort the sets in ascending order. Compare the first pail in each set and choose the set with the smallest pail. If the first pails match, compare the second pails and choose from among those, else continue until the two sets differ. Thus the set {3, 5, 7, 100} should be chosen over {3, 6, 7, 8}.

To measure out milk, FJ may completely fill a pail from the tank and pour it into the bottle. He can never remove milk from the bottle or pour milk anywhere except into the bottle. With a one-quart pail, FJ would need only one pail to create any number of quarts in a bottle. Other pail combinations are not so convenient.

Determine the optimally small number of pails to purchase, given the guarantee that at least one solution is possible for all contest input data.

PROGRAM NAME: milk4

INPUT FORMAT

Line 1:	The single integer Q
Line 2:	A single integer P (1 <= P <= 100) which is the number of pails in the store
Lines 3..P+2:	Each line contains a single integer pail_value (1 <= pail_value <= 10000), the number of quarts a pail holds

SAMPLE INPUT (file milk4.in)

```
16
3
3
5
7
```

OUTPUT FORMAT

The output is a single line of space separated integers that contains:

- the minimum number of pails required to measure out the desired number of quarts, followed by:
- a sorted list (from smallest to largest) of the capacity of each of the required pails

SAMPLE OUTPUT (file milk4.out)

```
2 3 5
```

5.3.2 Window Area

IV Balkan Olympiad

You've just be assigned the project of implemented a windowing interface. This windowing interface is fairly simple, and fortunately, you don't have to display the actual windows. There are 5 basic operations:

- Create a window
- Bring a window to the top
- Put a window to the bottom
- Destroy a window
- Output what percentage of a window is visible (i.e., isn't covered by windows above it).

In the input, the operations appear in the following format:

- Create window: w(I,x,y,X,Y)
- Bring window to top: t(I)
- Put window on bottom: b(I)
- Destroy window: d(I)
- Output percentage visible: s(I)

The I is a unique identifier for each window, which is one character. The character can be any of 'a'..'z', 'A'..'Z', and '0'..'9'. No extra spaces will appear in the input.

(x,y) and (X,Y) are opposite corners of the window. When a window is created, it is put `on top'. You can't create a window with an identifier that is already in use, but you can destroy a window and then create a new one with the identifier of the destroyed window. Coordinates will be positive integers, and all windows will be of non-zero area (x != X and y != Y). The x and y coordinates are between 1 and 32767 inclusive.

PROGRAM NAME: window

INPUT FORMAT

The input file consists of a sequence of commands to your interpreter. They will be listed one per line. Terminate the program when no more input is available

SAMPLE INPUT (file window.in)

```
w(a,10,132,20,12)
w(b,8,76,124,15)
s(a)
```

OUTPUT FORMAT

Output lines only for the s() commands. Of course, there might be several s() commands (but no more than 500) so the output should be a sequence of percentages, one per line, stating the percentage of the windows that are visible. The percentages should be rounded to 3 decimal places.

SAMPLE OUTPUT (file window.out)

```
49.167
```

5.3.3 Network of Schools

IOI '96 Day 1 Problem 3

A number of schools are connected to a computer network. Agreements have been developed among those schools: each school maintains a list of schools to which it distributes software (the "receiving schools"). Note that if B is in the distribution list of school A, then A does not necessarily appear in the list of school B.

You are to write a program that computes the minimal number of schools that must receive a copy of the new software in order for the software to reach all schools in the network according to the agreement (Subtask A). As a further task, we want to ensure that by sending the copy of new software to an arbitrary school, this software will reach all schools in the

network. To achieve this goal we may have to extend the lists of receivers by new members. Compute the minimal number of extensions that have to be made so that whatever school we send the new software to, it will reach all other schools (Subtask B). One extension means introducing one new member into the list of receivers of one school.

PROGRAM NAME: schlnet

INPUT FORMAT

The first line of the input file contains an integer N: the number of schools in the network (2<=N<=100). The schools are identified by the first N positive integers. Each of the next N lines describes a list of receivers. The line i+1 contains the identifiers of the receivers of school i. Each list ends with a 0. An empty list contains a 0 alone in the line.

SAMPLE INPUT (file schlnet.in)

```
5
2 4 3 0
4 5 0
0
0
1 0
```

OUTPUT FORMAT

Your program should write two lines to the output file. The first line should contain one positive integer: the solution of subtask A. The second line should contain the solution of subtask B.

SAMPLE OUTPUT (file schlnet.out)

```
1
2
```

5.3.4 Big Barn

A Special Treat

Farmer John wants to place a big square barn on his square farm. He hates to cut down trees on his farm and wants to find a location for his barn that enables him to build it only on land that is already clear of trees. For our purposes, his land is divided into N x N parcels. The input contains a list of parcels that contain trees. Your job is to determine and report the largest possible square barn that can be placed on his land without having to clear away trees. The barn sides must be parallel to the horizontal or vertical axis.

EXAMPLE

Consider the following grid of Farmer John's land where `.' represents a parcel with no trees and `#' represents a parcel with trees:

```
    1 2 3 4 5 6 7 8
  1 . . . . . . . .
  2 . # . . . # . .
  3 . . . . . . . .
  4 . . . . . . . .
  5 . . . . . . . .
  6 . . # . . . . .
  7 . . . . . . . .
  8 . . . . . . . .
```

The largest barn is 5 x 5 and can be placed in either of two locations in the lower right part of the grid.

PROGRAM NAME: bigbrn

INPUT FORMAT

Line 1: Two integers: N (1 <= N <= 1000), the number of parcels on a side, and T (1 <= T <= 10,000) the number of parcels with trees

Lines 2..T+1: Two integers (1 <= each integer <= N), the row and column of a tree parcel

SAMPLE INPUT (file bigbrn.in)

```
8 3
```

```
2 2
2 6
6 3
```

OUTPUT FORMAT

The output file should consist of exactly one line, the maximum side length of John's barn.

SAMPLE OUTPUT (file bigbrn.out)

```
5
```

Section 5.4

5.4.1 All Latin Squares

A square arrangement of numbers

```
1  2  3  4  5
2  1  4  5  3
3  4  5  1  2
4  5  2  3  1
5  3  1  2  4
```

is a 5 x 5 Latin Square because each whole number from 1 to 5 appears once and only once in each row and column.

Write a program that will compute the number of NxN Latin Squares whose first row is:

```
1 2 3 4 5.......N
```

Your program should work for any N from 2 to 7.

PROGRAM NAME: latin

INPUT FORMAT

One line containing the integer N.

SAMPLE INPUT (file latin.in)

```
5
```

OUTPUT FORMAT

A single integer telling the number of latin squares whose first row is 1 2 3 . . . N.

SAMPLE OUTPUT (file latin.out)

```
1344
```

5.4.2 Canada Tour

You have won a contest sponsored by an airline. The prize is a ticket to travel around Canada, beginning in the most western point served by this airline, then traveling only from west to east until you reach the most eastern point served, and then coming back only from east to west until you reach the starting city. No city may be visited more than once, except for the starting city, which must be visited exactly twice (at the beginning and the end of the trip). You are not allowed to use any other airline or any other means of transportation.

Given a list of cities served by the airline and a list of direct flights between pairs of cities, find an itinerary which visits as many cities as possible and satisfies the above conditions beginning with the first city and visiting the last city on the list and returning to the first city.

PROGRAM NAME: tour

INPUT FORMAT

Line 1:	The number N of cities served by the airline and the number V of direct flights that will be listed. N will be a positive integer not larger than 100. V is any positive integer.
Lines 2..N+1:	Each line contains a name of a city served by the airline. The names are ordered from west to east in the

input file. There are no two cities in the same meridian. The name of each city is a string of, at most, 15 digits and/or characters of the Latin alphabet; there are no spaces in the name of a city.

Lines N+2..N+2+V-1: Each line contains two names of cities (taken from the supplied list), separated by a single blank space. This pair is connected by a direct, two-way airline flight.

SAMPLE INPUT (file tour.in)

```
8 9
Vancouver
Yellowknife
Edmonton
Calgary
Winnipeg
Toronto
Montreal
Halifax
Vancouver Edmonton
Vancouver Calgary
Calgary Winnipeg
Winnipeg Toronto
Toronto Halifax
Montreal Halifax
Edmonton Montreal
Edmonton Yellowknife
Edmonton Calgary
```

OUTPUT FORMAT

Line 1: The number M of different cities visited in the optimal itinerary. Output 1 if no itinerary is possible.

SAMPLE OUTPUT (file tour.out)

```
7
```

Namely: Vancouver, Edmonton, Montreal, Halifax, Toronto, Winnipeg, Calgary, and Vancouver (but that's not a different city).

5.4.3 Character Recognition

This problem requires you to write a program that performs character recognition.

Each ideal character image has 20 lines of 20 digits. Each digit is a `0' or a `1'. See Figure 1a (way below) for the layout of character images in the file.

The file font.in[iii] contains representations of 27 ideal character images in this order:

```
_abcdefghijklmnopqrstuvwxyz
```

where _ represents the space character. Each ideal character is 20 lines long.

The input file contains one or more potentially corrupted character images. A character image might be corrupted in these ways:

- at most one line might be duplicated (and the duplicate immediately follows)
- at most one line might be missing
- some 0's might be changed to 1's
- some 1's might be changed to 0's.

No character image will have both a duplicated line **and** a missing line. No more than 30% of the 0's and 1's will be changed in any character image in the evaluation datasets.

In the case of a duplicated line, one or both of the resulting lines may have corruptions, and the corruptions may be different.

[iii] See http://ace.delos.com/usaco/font.in.

Write a program to recognize the sequence of one or more characters in the image provided in the input file using the font provided in file font.in.

Recognize a character image by choosing the font character images that require the smallest number of overall changed 1's and 0's to be corrupted to the given font image, given the most favourable assumptions about duplicated or omitted lines. Count corruptions in only the least corrupted line in the case of a duplicated line. You must determine the sequence of characters that most closely matches the input sequence (the one that requires the least number of corruptions). There is a unique best solution for each evaluation dataset.

A correct solution will use precisely all of the data supplied in the input file.

PROGRAM NAME: charrec

INPUT FORMAT (both input files)

Both input files begin with an integer N (19 <= N < 1200) that specifies the number of lines that follow:

```
N
(digit1)(digit2)(digit3) ... (digit20)
(digit1)(digit2)(digit3) ... (digit20)
...
```

Each line of data is 20 digits wide. There are no spaces separating the zeros and ones.

The file font.in describes the font. It will always contain 541 lines. It may differ for each evaluation dataset.

SAMPLE INPUT (file charrec.in)

Incomplete sample showing the beginning of font.in (space and a).

font.in

```
540
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000011100000000000
00000111111011000000
00001111111001100000
00001110001100100000
00001100001100010000
00001100000100010000
00000100000100010000
00000010000000110000
00000001000001110000
00001111111111110000
00001111111111110000
```

Sample charrec.in, showing an a corrupted

charrec.in

```
19
00000000000000000000
00000000000000000000
00000000000000000000
00000011100000000000
00100111011011000000
00001111111001100000
00001110001100100000
00001100001100010000
00001100000100010000
00000100000100010000
00000010000000110000
00001111011111110000
00001111111111110000
00001111111111000000
00001000010000000000
00000000000000000000
00000000000001000000
00000000000000000000
00000000000000000000
```

```
00001111111111000000
00001000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
```

Figure 1a *Figure 1b*

OUTPUT FORMAT

Your program must produce an output file that contains a single string of the characters recognized. Its format is a single line of ASCII text. The output should not contain any separator characters. If your program does not recognize a particular character, it must output a ? in the appropriate position.

SAMPLE OUTPUT (file charrec.out)

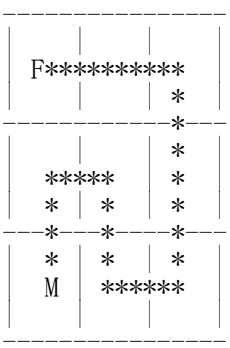
```
 a
```

Note that the output is a line with two characters: a blank followed by an `a'.

5.4.4 Betsy's Tour

Don Piele

A square township has been divided up into N^2 square plots (1 <= N <= 7). The Farm is located in the upper left plot and the Market is located in the lower left plot. Betsy takes her tour of the township going from Farm to Market by walking through every plot exactly once. Shown below is one possible tour for Betsy when N=3.



Write a program that will count how many unique tours Betsy can take in going from Farm to Market for any value of N.

PROGRAM NAME: betsy

INPUT FORMAT

Line 1: A single integer N (1 <= N <= 7)

SAMPLE INPUT (file betsy.in)

```
3
```

OUTPUT FORMAT

A single line with a single integer, the number of unique tours.

SAMPLE OUTPUT (file betsy.out)

```
2
```

5.4.5 Telecowmunication

Farmer John's cows like to keep in touch via email so they have created a network of cowputers so that they can intercowmunicate. These machines route email so that if there exists a sequence of c cowputers a1, a2, ..., a(c) such that a1 is connected to a2, a2 is connected to a3, and so on then a1 and a(c) can send email to one another.

Unfortunately, a cow will occasionally step on a cowputer or Farmer John will drive over it, and the machine will stop working. This means that the cowputer can no longer route email, so connections to and from that cowputer are no longer

usable.

Two cows are pondering the minimum number of these accidents that can occur before they can no longer use their two favorite cowputers to send email to each other. Write a program to calculate this minimal value for them, and to calculate a set of machines that corresponds to this minimum.

For example the network:

```
  1*
 /
3 - 2*
```

shows 3 cowputers connected with 2 lines. We want to send messages between 1 with 2. Direct lines connect 1-3 and 2-3. If cowputer 3 is down, them there is no way to get a message from 1 to 2.

PROGRAM NAME: telecow

INPUT FORMAT

Line 1	Four space-separated integers: N, M, c1, and c2. N is the number of computers (1 <= N <= 100), which are numbered 1..N. M is the number of connections between pairs of cowputers (1 <= M <= 600). The last two numbers, c1 and c2, are the id numbers of the cowputers that the questioning cows are using. Each connection is unique and bidirectional (if c1 is connected to c2, then c2 is connected to c1). There can be at most one wire between any two given cowputers. Computer c1 and c2 will not have a direction connection.
Lines 2..M+1	The subsequent M lines contain pairs of cowputers id numbers that have connections between them.

SAMPLE INPUT (file telecow.in)

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

OUTPUT FORMAT

Generate two lines of output. The first line is the minimum number of cowputers that can be down before terminals c1 & c2 are no longer connected. The second line is a minimal-length sorted list of cowputers that will cause c1 & c2 to no longer be connected. Note that neither c1 nor c2 can go down. In case of ties, the program should output the set of computers that, if interpreted as a base N number, is the smallest one.

SAMPLE OUTPUT (file telecow.out)

```
1
3
```

Section 5.5

5.5.1 Picture

IOI 1998

A number, N (1 <= N < 5000), of rectangular posters, photographs and other pictures of the same shape are pasted on a wall. Their sides are all vertical or horizontal. Each rectangle can be partially or totally covered by the others. The length of the boundary of the union of all rectangles is called the perimeter. Write a program to calculate the perimeter.

Figure 1 shows an example with seven rectangles:

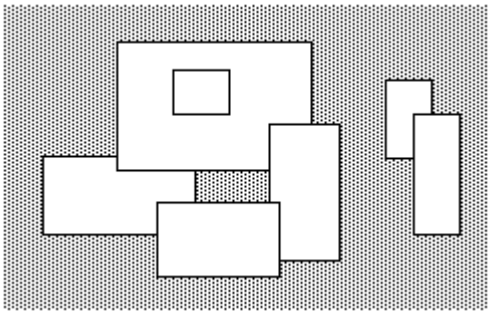


Figure 1. A set of seven rectangles

The corresponding boundary is the whole set of line segments drawn in Figure 2:

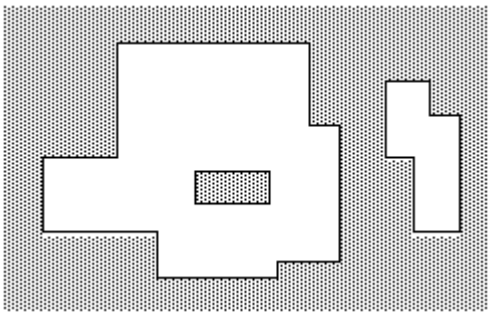


Figure 2. The boundary of the set of rectangles

The vertices of all rectangles have integer coordinates. All coordinates are in the range [-10000,10000] and any existing rectangle has a positive area. The numeric value of the result fits in a 32-bit signed representation.

PROGRAM NAME: picture

INPUT FORMAT

Line 1:	N, the number of rectangles pasted on the wall.
Lines 2..N+1	In each of the subsequent lines, one can find the integer coordinates of the lower left vertex and the upper right vertex of each rectangle. The values of those coordinates are given as ordered pairs consisting of an x-coordinate followed by a y-coordinate.

SAMPLE INPUT (file picture.in)

```
7
-15 0 5 10
-5 8 20 25
15 -4 24 14
0 -6 16 4
2 15 10 22
30 10 36 20
34 0 40 16
```

OUTPUT FORMAT

A single line with a non-negative integer which corresponds to the perimeter for the input rectangles.

SAMPLE OUTPUT (file picture.out)

```
228
```

5.5.2 Hidden Password

ACM South Eastern Europe -- 2003

Sometimes the programmers have very strange ways of hiding their passwords. Billy "Hacker" Geits chooses a string S composed of L (5 <= L <= 100,000) lowercase letters ('a'..'z') with length L. Then he makes and sorts all L-1 one-letter left cyclic shifts of the string. He then takes as a password one prefix of the lexicographically first of the obtained strings (including S).

For example consider the string "alabala". The sorted cyclic one-letter left shifts (including the initial string) are:

```
aalabal
abalaal
alaalab
alabala
balaala
laalaba
labalaa
```

Lexicographically, first string is 'aalabal'. The first letter of this string ('a') is the 'a' that was in position 6 in the initial string (counting the first letter in the string as position 0).

Write a program that, for given string S, finds the start position of the first letter of the sorted list of cyclic shifts of the string. If the first element appears more than once in the sorted list, then the program should output the smallest possible initial position.

PROGRAM NAME: hidden

INPUT FORMAT

- Line 1: A single integer: L
- Line 2..?: All L characters of the the string S, broken across lines such that each line has 72 characters except the last one, which might have fewer.

SAMPLE INPUT (file hidden.in)

```
7
alabala
```

OUTPUT FORMAT

- Line 1: A single integer that is the start position of the first letter, as described above.

SAMPLE OUTPUT (file hidden.out)

```
6
```

5.5.3 Towfive

IOI 2001

In order to teach her young calvess the order of the letters in the alphabet, Bessie has come up with a game to play with them. The calves are given a 5 x 5 grid on which they can put the letters 'A'-'Y', but the one rule is that all the letters going across the columns and down the rows must be in the order they appear in the alphabet.

There are a huge number of possible grids, so Bessie decides that they will be named by the string of letters that is given by reading across each row, going down the rows. For example, the grid:

```
A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
```

would have the name ABCDEFGHIJKLMNOPQRSTUVWXY, which is coincidentally the first possible grid when the entire set of grids is ordered alphabetically. The second grid that meets this requirement is ABCDEFGHIJKLMNOPQRSUTVWXY, which is formed by switching the 'T' and 'U' in the above grid.

Help the calves gain bragging rights. Given a number, M, find which string is Mth in the list of all possible grids when they are sorted alphabetically, and, given a string of letters, find out what number the corresponding grid is in the list of all possible grids.

PROGRAM NAME: twofive

INPUT FORMAT

The first input line contains one of two letters, either an 'N' or a 'W'.

If the first input line contains an 'N', the second line will contain an integer, M, that corresponds to the number of a valid grid. If the first line contains a 'W', the second line will contain a string of 25 letters, which represents a valid grid.

OUTPUT FORMAT

If the input contained the number of a valid grid (first line 'N'), the output should contain a string of 25 letters on a line, which corresponds to the Mth grid in the sorted list of all possible grids.

If the input contained a string of letters indicating a grid (first line 'W'), the output should contain a single integer on a line, which corresponds to the number of the given grid in the list of all possible grids.

SAMPLE INPUT #1 (file twofive.in)

```
N
2
```

SAMPLE OUTPUT #1 (file twofive.out)

```
ABCDEFGHIJKLMNOPQRSUTVWXY
```

SAMPLE INPUT #2 (file twofive.in)

```
W
ABCDEFGHIJKLMNOPQRSUTVWXY
```

SAMPLE OUTPUT #2 (file twofive.out)

```
2
```

Chapter 6 Contest Practice

Section 6.1

6.1.1 Postal Vans

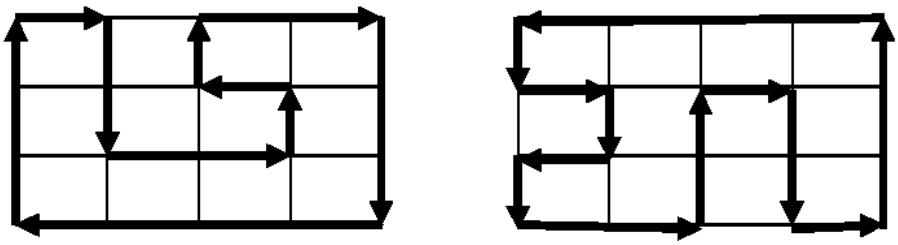
ACM South Pacific Region -- 2003

Tiring of their idyllic fields, the cows have moved to a new suburb. The suburb is a rectangular grid of streets with a post office at its Northwest corner. It has four avenues running East-West and N (1 <= N <= 1000) streets running North-South.

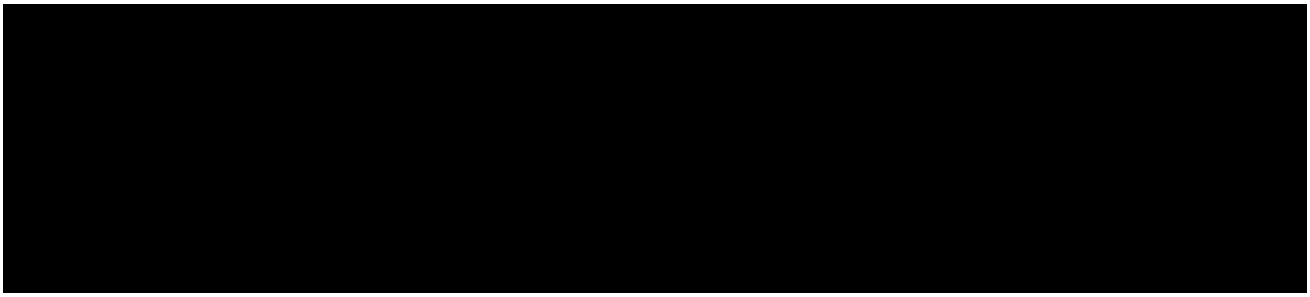
For example, the following diagram shows such a suburb with N=5 streets, with the avenues depicted as horizontal lines, and the post office as a dark blob at the top-left corner:

Each day the postal van leaves the post office, drives around the suburb and returns to the post office, passing exactly once through every intersection (including those on borders or corners). The executives from the post company want to know how many distinct routes can be established for the postal van (of course, the route direction is significant in this count).

For example, the following diagrams show two such routes for the above suburb:



As another example, the following diagrams show all the four possible routes for a suburb with N=3 streets:



Write a program that will determine the number of such distinct routes given the number of streets.

PROGRAM NAME: vans

INPUT FORMAT

- Line 1: A single integer, N

SAMPLE INPUT (file vans.in)

```
4
```

OUTPUT FORMAT

- Line 1: A single integer that tells how many possible distinct routes corresponding to the number of streets given in

the input.

SAMPLE OUTPUT (file vans.out)

```
12
```

6.1.2 A Rectangular Barn

Mircea Pasoi -- 2003

Ever the capitalist, Farmer John wants to extend his milking business by purchasing more cows. He needs space to build a new barn for the cows.

FJ purchased a rectangular field with R ($1 \le R \le 3,000$) rows numbered 1..R and C ($1 \le C \le 3,000$) columns numbered 1..C. Unfortunately, he realized too late that some 1x1 areas in the field are damaged, so he cannot build the barn on the entire RxC field.

FJ has counted P ($0 \le P \le 30,000$) damaged 1x1 pieces and has asked for your help to find the biggest rectangular barn (i.e., the largest area) that he can build on his land without building on the damaged pieces.

PROGRAM NAME: rectbarn

INPUT FORMAT

- Line 1: Three space-separated integers: R, C, and P.
- Lines 2..P+1: Each line contains two space-separated integers, r and c, that give the row and column numbers of a damaged area of the field

SAMPLE INPUT (file rectbarn.in)

```
3 4 2
1 3
2 1
```

OUTPUT FORMAT

- Line 1: The largest possible area of the new barn

SAMPLE OUTPUT (file rectbarn.out)

```
6
```

OUTPUT DETAILS

```
  1 2 3 4
 +-+-+-+-+
1| | |X| |
 +-+-+-+-+
2|X|#|#|#|
 +-+-+-+-+
3| |#|#|#|
 +-+-+-+-+
```

Pieces marked with 'X' are damaged and pieces marked with '#' are part of the new barn.

6.1.3 Cow XOR

Adrian Vladu -- 2005

Farmer John is stuck with another problem while feeding his cows. All of his N ($1 \le N \le 100,000$) cows (numbered 1..N) are lined up in front of the barn, sorted by their rank in their social hierarchy. Cow #1 has the highest rank; cow #N has the least rank. Every cow had additionally been assigned a non-unique integer number in the range $0..(2^{21} - 1)$.

Help FJ choose which cows will be fed first by selecting a sequence of consecutive cows in the line such that the bitwise "xor" between their assigned numbers has the maximum value. If there are several such sequences, choose the sequence for which its last cow has the highest rank. If there still is a tie, choose the shortest sequence.

TIME LIMIT: 0.5 sec

PROGRAM NAME: cowxor

INPUT FORMAT

- Line 1: A single integer N
- Lines 2..N+1: N integers ranging from 0 to 2^{21} - 1, representing the cows' assigned numbers. Line j describes cow of social hierarchy j-1.

SAMPLE INPUT (file cowxor.in)

```
5
1
0
5
4
2
```

INPUT DETAILS:

There are 5 cows. Cow #1 had been assigned with 1; cow #2 with 0; cow #3 with 5; cow #4 with 4; cow #5 with 2.

OUTPUT FORMAT

- Line 1: Three space-separated integers, respectively: the maximum requested value, the position where the sequence begins, the position where the sequence ends.

SAMPLE OUTPUT (file cowxor.out)

```
6 4 5
```

OUTPUT DETAILS:

4 xor 2 = 6 (001) xor (010) = (011)

Appendices

Hints (use them carefully!)

Section 1.5.2 Prime Palindromes

HINT 1: Generate the palindromes and see if they are prime.

HINT 2: Generate palindromes by combining digits properly. You might need more than one of the loops like below.

```
/* generate five digit palindrome: */
for (d1 = 1; d1 <= 9; d1+=2) {  /* only odd; evens aren't so prime */
    for (d2 = 0; d2 <= 9; d2++) {
        for (d3 = 0; d3 <= 9; d3++) {
        palindrome = 10000*d1 + 1000*d2 +100*d3 + 10*d2 + d1;
        ... deal with palindrome ...
    }
    }
}
```

Section 1.5.4 Checker Challenge

HINT 1:

Use recursion:

```
    function placequeen(column) {   # place columns 0..max-1
    if (column == max) { deal with answer; return; }
        for (row = 0; row < max; row++)  {
            if (canplacequeen (row)) {
        mark queen placed at column,row;
        placequeen(column+1);
        un-mark queen placed at column,row;
        }
        }
    }
```

HINT 2:

Do everything you can to eliminate loops (searching) in the part of your program executed most frequently. Usually, the best way to do this is to 'trade memory for speed'. When checking to see if a queen can be placed in a given row, for example, store a row status list that says if a queen is already stored there or not:

```
    function placequeen(column) {   # place columns 0..max-1
    if (column == max) { deal with answer; return; }
        for (row = 0; row < max; row++)  {
            if (rowok[row] && canplacequeen(row,column)) {
        rowok[row] = 1;
        mark queen placed at column,row;
        placequeen(column+1);
        un-mark queen placed at column,row;
        rowok[row] = 0;
        }
        }
    }
```

HINT 3:

Do *absolutely everything* you can to eliminate loops (searching) in the part of your program executed most frequently. Keep track not only of the rows that are legal for queen placement but also which of the two sorts of diagonals (the ones that are like a '/' and the others that are like '\' by using an array of size 2*max - 1 that records whether a diagonal is legal or not.

HINT 4:

SYMMETRY. Can you eliminate half or 3/4 of the cases you test by studying rotations, reflections, or something like that? *[hint: yes]*

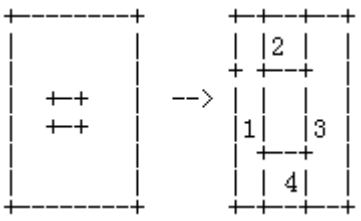
HINT 5:

Still over time? If you have programmed modularly and have little subroutines to check diagonals, etc., move that code into the main execution stream. The subroutine call overhead is nontrivial.

Section 3.1.4 Shaping Regions

An array of all 'points' is too big; 16MB maximum.

Keep track of the rectangles' coordinates; split the rectangle when an overlap occurs, e.g.:



Section 4.1.2 Fence Rails

This is a high dimensionality multiple knapsack problem, so we just have to test the cases. Given that the search space has a high out-degree, we will use depth first search with iterative deepening in order to limit the depth of the tree. However, straight DFSID will be too slow, so some tree-pruning is necessary.

Submitting Solutions

The restrictions are few:

- One second runtime limit unless other specified (700 MHz Pentium III)
- About 16MB datasize limit
- About 1MB stacksize limit
- Be sure your program exits with status 0
- Be sure you print complete lines (with terminating newline), not just a few words or numbers
- Don't use files other than the specified input, output, and auxiliary files
- Other common sense rules that need not be listed

The rules are simple:

- Don't try to cheat.
- **Don't just print the answers**, you must calculate them in your program. If you just print answers, your login ID might be removed.
- Don't try to look at other files on the system or use other schemes to break security
- Don't try to break common sense rules of privacy
- Please report anomalous behavior to me right away (mailto:kolstad@ace.delos.com)
- Have as much fun as possible
- Earn a trip to the IOI and other exotic contests!

Some hints:

- Both stderr and stdout are returned to you when errors occur
- Feel free to ask questions and send in comments
- Your reported output has `_'s substituted for spaces
- Include this comment if you use try/catch/throw in C++: /*pragma handle-exceptions*/

Compiler comments (please send in new compiler comments as you find them):

- We're using g++ (a.k.a. djgpp on PCs), Free Pascal, and gjc
- In C/C++, ints are 32 bits (char is 8; short is 16; long is 32; long long is 64)
- some libraries have new names; some have different or missing functions
- stricmp doesn't exist; use strcmp for string compares
- strrev does not exist
- neither itoa nor ltoa exists (use sprintf instead)
- No need for *huge* declarations - pointers already go everywhere
- Pascal users: be sure to "close" your output file or the output might not appear

Words

pasture	牧场
trail	小路
hay	干草

barn	牛棚
intersection	十字路口

grazing location	放牧地点
hoof	牛蹄（复hooves）

FJ：农夫约翰（Farmer John）的缩写。

Bessie：贝西，一头非常聪明的母牛。