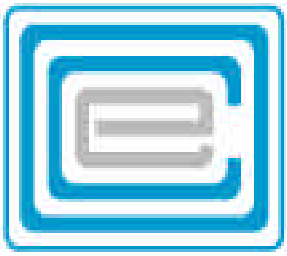


ECOO 2003
Programming Contest

Board wide Contest

to be written after March 2rd

Problem 1- Center of gravity

Find the center of gravity among a set of up to 10 spheres

DATA11 (DATA12 for the second try) contains 5 sets of data in the form of positive integers between 1 and 10000, each integer on a separate line. The first line of each data set contains n, the number of spheres in the system (a number between 1 and 10). Then follows n sets of 4 numbers: the x, y, z value for the center of mass of each sphere and m, its mass.

Write a program that will calculate the center of gravity of each system to 1 digit accuracy after the decimal point.

input:
(to save space each set is in a separate column. However, in the input file, each set immediately follows the previous set)

```
1          5          4          5          6
32         25         50         21         62
31         34         21         60         30
60         82         53         33         27
1038       1008       1008       504        840
           39         50         21         83
           44         21         52         45
           85         59         26         27
           1008       1260       189        105
           31         54         41         75
           52         51         76         49
           83         62         46         47
           720        378        175        21
           65         62         53         91
           57         55         89         53
           109        72         45         57
           240        378        70         30
           71                    83         117
           69                    99         58
           121                   65         83
           48                    70         8
                                            123
                                            64
                                            89
                                            4
```

output:

```
system 1 is centered about (32.0 , 31.0 , 60.0) with mass 1038.0
system 2 is centered about (35.0 , 44.0 , 86.0) with mass 3024.0
system 3 is centered about (52.0 , 29.0 , 59.0) with mass 3024.0
system 4 is centered about (31.0 , 66.0 , 37.0) with mass 1008.0
system 5 is centered about (66.0 , 33.0 , 29.0) with mass 1008.0
```

Problem 2 - Ancestors

Genealogists have an interesting way of ordering their ancestors. The SELF is considered number 1. Father is 2 and mother is 3. There are 4 grandparents in the third generation, numbered 4,5,6 and 7. The 8 great-grandparents are numbered 8, 9, 10, 11, 12, 13, 14 and 15. It is not so confusing, if you consider that the father of number x is 2x, and the mother of x is 2x+1. Since one's father is 2, then the father of one's father is 2(2) and his mother: 2(2)+1 = 5.
The father of 345 is of course 690 and her mother is 691.

Write a program that lets you identify the ancestor by generic name:
In generation 1 there is one person: Susan (the "self")
In generation 2 there are 2 people: father, mother
In generation 3 there are 4 people: 2 grandfathers, 2 grandmothers
In generation 4 there are 8 people: 4 great-grandfathers, 4 great-grandmothers
in generation 5 there are 16 people: great-2-grandfathers and great-2-grandmothers
and so on
in general, if x>1, then in generation x+3,the people are named great-x-grandfathers and great-x-grandmothers.

The first 128 lines of the file contain family names. They are the last names of the 256 ancestors of generation 9 in numeric order: The first couple (#256, #257) are named Adams, the next couple, (#258, #259) are called Alexander, and so on. The last couple (#510, #511) have the family name of Perrault.

Great-6-grandfather Adams (#256) and great-6-grandmother Adams (#257) are the parents of great-5-grandfather Adams (#128).

Great-6-grandfather Alexander (#258) and great-6-grandmother Alexander (#259) are the parents of great-5-grandmother Adams (#129) Not her maiden name of course.

Note that both Susan's father and Susan's mother are called: father Adams and mother Adams. Note that Susan's mother's parents take the 65th name in the list: they are grandfather Griffin (#6) and grandmother Griffin (#7)

Both DATA21 and DATA22 contain 128 last names, followed by 10 numbers between 1 and 511, representing 10 ancestors of Susan Adams, or Susan herself.
You must identify these ancestors as in the sample solution on the next page.

Note: Since there will be 10 lines in your solution, each correct line will earn you 10 points.

To save space, the data is presented as shown. The text file however will have this data listed on 128+10 =138 lines.

sample input: (the 128 names)

Adams	Boucher	Cooper	Fortin	Griffin	James	Lopez	Murphy
Alexander	Brooks	Cote	Foster	Griffiths	Jenkins	Marshall	Murray
Allard	Brown	Cox	Fournier	Hall	Johnson	Martel	Nelson
Allen	Bryant	Davies	Gagne	Harris	Jones	Martin	Ouellet
Anderson	Butler	Davis	Gagnon	Harrison	Kelly	Martin	Owen
Archambault	Campbell	Demers	Garcia	Hayes	King	Martinez	Palmer
Bailey	Caron	Diaz	Gauthier	Hebert	Knight	Mason	Paquet
Baker	Carter	Dixon	Gibson	Henderson	Langlois	Matthews	Parent
Barker	Chapman	Dubois	Girard	Hernandez	Leclerc	Menard	Parker
Barnes	Charbonneau	Dupuy	Giroux	Hill	Leduc	Miller	Patel
Baudouin	Clark	Edwards	Gonzales	Holmes	Lee	Mills	Patterson
Bedard	Clarke	Ellis	Gonzalez	Houde	Lefebvre	Mitchell	Pearson
Belanger	Cloutier	Evans	Gosselin	Howard	Levesque	Moore	Pelletier
Bell	Coleman	Fisher	Graham	Hughes	Lewis	Morgan	Pepin
Bennett	Collins	Flores	Gray	Hunt	Lloyd	Morin	Perez
Bouchard	Cook	Fortier	Green	Jackson	Long	Morris	Perrault

sample input: (the 10 numbers)

```
3
377
500
1
18
100
321
25
7
15
```

sample output:

```
  3 represents: mother Adams
377 represents: great-6-grandmother Gosselin
500 represents: great-6-grandfather Patterson
  1 represents: Susan Adams
 18 represents: great-2-grandfather Boucher
100 represents: great-4-grandfather Hernandez
321 represents: great-6-grandmother Cooper
 25 represents: great-2-grandmother Griffin
  7 represents: grandmother Griffin
 15 represents: great-grandmother Lopez
```

Problem 3- Balloonist

In order to travel in a certain direction, a balloonist must rise or descend to the layer of air where the wind blows in that direction. Today the balloonist finds herself in an awkward position. She has thrown out all her ballast, so she can no longer rise to a higher level. She can stay at a given height, or descend. The computer on board can tell her exactly what her position is (x,y) relative to home (0,0). If x is positive, she is north of home, if negative, south. If y is positive she is to the east of home, if negative, west.

The computer also informs her of the wind direction and wind speed of all the layers of air below her. Once she has arrived in a certain layer of air and is therefore traveling with the current wind, she can stay there indefinitely. Unfortunately, when descending through any given layer, it takes a minimum of 1 minute to pass through it, so that she may be blown off course during that minute. She must therefore carefully plot a path home.

In the first example there are 6 layers of air below her. She is located 100 km north of home and 50 km east, for the computer indicates that she is at (100,50). The winds in the six layers of air she must go through are respectively: S 11, E 4, N 18, E 10, W 5, S 20. This means that the first layer is moving in a southerly direction at 11 km per hour, the next layer is moving in an easterly direction at 4 km per hour and so on.

She wants to travel as fast as possible, and since the last two layers go in the right direction and are fastest, she will only spend the minimum 4 minutes traveling through the first 4 layers.

After passing through the first layer she finds herself at (99.82, 50). After the next layer she is at (99.82, 50.07) and after the third layer at (100.12, 50.23)
She must therefore stay in the 5th layer for 602.8 minutes and in the 6th layer for 300.35 minutes. Her total traveling time would therefore be 907.15 minutes, which may be rounded off to 907 minutes.

DATA31 (DATA32 for the second try) contains 5 sets of data. The first two lines in each set contains the x and y value of the initial position of the balloon, a number between -2000 and 2000. The third line in each set contains the number of layers of air the balloonist must travel through. This number, n, is between 4 and 20 inclusive. Then follow n lines containing one letter: N, E, W or S indicating the wind direction, one space character and an integer between 1 and 100, the speed of the wind in km per hour.

Write a program that will read the data, find the fastest way home and print out the amount of time this takes to the nearest minute. You may assume that in all cases each of the 4 directions are represented at least once, and so there is always a way to get home.

input
(in order to save space, the data is presented in 5 separate columns. On file the data would be in one column, taking up 42 lines.

100	1	1000	100	0
50	1	4	100	100
6	4	4	5	8
S 11	E 9	N 10	N 20	E 19
E 4	W 6	W 18	S 8	S 21
N 18	N 16	E 4	E 8	N 10
E 10	S 99	S 6	W 20	S 15
W 5			E 4	S 20
S 20				W 19
				E 11
				W 7

output

The balloon took 907 minutes to get home

The balloon took 16 minutes to get home

The balloon took 10017 minutes to get home

The balloon took 1056 minutes to get home

The balloon took 329 minutes to get home

Problem 4 - RoboCar

Stan built a robot car that can drive along a straight runway and that can modify its speed every second: It can however only change its speed up or down by one meter per second: If it is going 5 meters per second during the space of one second, it can slow down to 4 meters per second, stay at 5 meters per second or speed up to 6 meters per second in the next time span. Stan can control its top speed as well as the total distance the car must travel.

Speed here is taken to mean **average** speed. To say, for example, that the speed at a given time is 7 meters per second, means, that during the given time span of one second, the car travels exactly 7 meters.

Stan's car will always travel as fast as possible, given that it must start from standstill and can't crash through the finish line: The finish line might possibly be a blank wall. During the first second the car's speed is therefore **one** meter per second, and during the last second, its speed must also be **one** meter per second.

If for example the car has been given the instruction to go 100 meters at a top speed of 8 meters per second, its best time will be 20 seconds, although the details of where it sped up or slowed down might vary. One solution is shown here:

elapsed seconds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
speed per second	1	2	3	4	5	6	7	8	8	7	7	7	7	7	6	5	4	3	2	1
kms travelled	1	3	6	10	15	21	28	36	44	51	58	65	72	79	85	90	94	97	99	100

DATA41 (DATA42 on the second try, will contain 10 lines each containing one integer. (Value between 1 and 32000 inclusive). They represent 5 sets of 2 data: First, the distance to be traveled and second, the speed limit of the robot car.

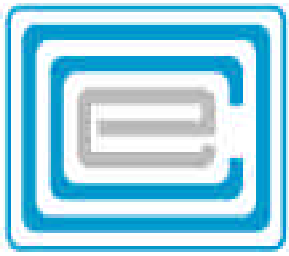
Write a program that will calculate the fastest time it will take to finish the race in each of the 5 instances. The output should mirror the output of the sample.

sample input:

```
100
8
1
10
43
1
1000
1000
32000
100
```

sample output:

```
Stan's car travels the 100 meters in 20 seconds
Stan's car travels the 1 meters in 1 seconds
Stan's car travels the 43 meters in 43 seconds
Stan's car travels the 1000 meters in 63 seconds
Stan's car travels the 32000 meters in 419 seconds
```



ECOO 2003

Nineteenth Annual Programming Contest

Regional

April 5 2003

Instructions to Programmers

1. You may use one computer and no printer. You may have the use of a calculator.

2. You may use any programming language, however it must be able to read a text file, and the output must be on the screen. You may use no other software or special libraries.

3. You may use one resource book, one that accompanies the programming software, but no third party manual is allowed.

4. Set up your machine before 11:00 a.m.
   The contest will take place between 11:00 a.m. and 2:00 p.m.

5. You have no more that 2 chances to have your program tested. Input data for the first try is DATAx1 and input data for the second try is DATAx2, where x = 1,2,3 or 4 the problem number.

6. Keep the score sheet near your machine.
   When you are ready to have your program tested, this is the procedure you should follow:
   - Wave the colour coded sheet in the air, to signal the appropriate judge that your program is ready to be tested.
   - The judge will bring over the disk. While two judges are present, copy the file onto your hard drive and RETURN the disk
   - YOU run the program: The judges are there to observe.
   - They will tally your score and you move on the next program. They will take your sheet with them to the Scoring station. It will be returned after it is processed.
   - Once the two judges move away from your machine, their decision, right or wrong, will be final. While deliberating, they may call over a 3rd person to arbitrate: If two of the three agree, that decision is final.

7. At 2:00 p.m. the contest ends, and many teams will probably wish to take the opportunity to have their program judged for the last time. Therefore DURING THE LAST FIVE MINUTES OF THE CONTEST, only step 1 of the judging process will take place, namely the jotting down of the time. The other steps can take place at leisure after 2:00. Therefore, at 2:00 pm stand well away from your machine, to clearly indicate, that you are no longer working on it.

Problem 1 - Base Three Code

All the capital letters, together with the space character make 27 distinct characters, and 27 is a power of 3. This fact forms the basis for the following secret code:

1- Change the letters in the message to UPPER CASE.

2- Eliminate all characters that are not one of the 27 characters allowed, substituting the * character for “space”

3- Express each character as a 3-digit base 3 number: A becomes 000.
B, C, D and E become 001, 002, 010, 011 respectively and so on:
Z becomes 221 and the space character becomes 222.

4- If the original length of the string was n characters, it now is 3n digits long.
Split the new string into 3 parts, each n digits long: part A, part B and part C.

5- Form a new set of 3-digit numbers, where the first digit comes from part A; the second digit comes from part B and the last digit comes from part C.

6- Convert each 3-digit number back to one of the 27 characters, a capital letter or a star (*) character.

A message such as: `I am the captain`
first becomes:
`I*AM*THE*CAPTAIN`

and then gets translated to the base 3 numbers:
`022-222-000-110-222-201-021-011-222-002-000-120-201-000-022-111`

then from the three groups;
`0222220001102222  0102101122200200  0120201000022111`

it becomes:
`000-211-202-220-212-200-011-010-020-120-120-002-202-221-201-201`

and finally:
AWUYXSEDGPPCUZTT

DATA11 (DATA12 for the second try) contains data on 5 separate lines. Each line contains the secret code of one message in the form of a string of characters. No line is longer than 85 characters.

Write a program that decodes each message and prints out the five messages on the screen, as in the example below:

Sample input:

```
CWYTZUFCBLJG*XWWHPGVMHLNZ*ZIISEB
GEIJOPHPAYSXFHEW*UOUYHZTCZBC
FXXY**IAIPQHXTTXMEG
HBHRRLDPIYZTHFNELMTS
G*YTTUHH*MQYLRIJPCBIFPKNIQH
```

Sample output:

```
I AM THE CAPTAIN OF THE PINAFORE
AND A RIGHT GOOD CAPTAIN TOO
I AM VERY VERY GOOD
AND BE IT UNDERSTOOD
I COMMAND A RIGHT GOOD CREW
```

(words from HMS Pinafore by Gilbert and Sullivan)

Problem 2 - Web Sorting

Imagine a set of communities in some fictional land, that wanted to be connected to the internet. Each community would be connected to a minimum of two other communities by cabling. The idea is, of course that in case one connection failed, they would always be able to count on the other connection.

They hit upon the following idea: To find the one community whose distance to all the other communities was on the average the shortest. It would be called THE CAPITAL. The capital would get a direct connection to the Internet. All others would have to connect to the capital. For a second connection, all the other towns would have to pick a partner town and they would connect to each other. Since there were an even number of towns, excluding the capital, this could easily be done.

It is not practical to find the one solution to partnering communities that would result in the least amount of cabling. However, the following strategy would come close:
Of all the remaining towns, pick the two towns that are the closest, and connect these. Then of the remaining towns, pick the two towns that are the closest, and connect these. Repeat this procedure, until all towns are partnered off.

DATA21 (DATA22 for the second try) contains the 5 sets of data. The first line in each set contains an odd integer, n, between 3 and 25, representing the number of communities under consideration. Then follow n sets of 2 integers, which stand for the x and y values of the ordered pairs, representing the relative positions of each community in kilometers (they are positive integers less than 32000)
The towns are named in order by the corresponding letters of the alphabet.

Your task is to identify the capital in each set, (there is always one unique solution) and the list of pairs of towns that are partnered, as in the example below.

Note that the pairs of towns should be printed in the order in which you find each pair. And each pair should occur in alphabetical order:
The distances for example in the first sample output are:
For **BI** 342.2 km, for **DE** 461.3 km, for **CF** 538.2 km, for **AG** 2272.7 km and for **JK** 3073.5 km

Sample input: (for convenience each set is in a separate column, however, in the input file, each number occupies one separate line:

11	3	9	7	5
325	3880	845	380	1285
4400	255	2255	4610	4725
675	4870	3990	215	4255
3330	4265	2925	2365	4750
2340	4470	3815	3455	390
235	200	4475	2945	1535
2575		680	1965	1140
3545		1880	1735	1365
2540		4910	4080	4200
3085		1410	3800	15
2650		1910	50	
675		4125	4745	
2595		3360	4770	
4510		200	1515	
1935		2165		
3340		300		
365		1930		
3185		1240		
175				
1290				
2985				
45				

Sample output:

```
the Capital is: H
the partners are: BI DE CF AG JK

the Capital is: C
the partners are: AB

the Capital is: I
the partners are: AD GH BC EF

the Capital is: C
the partners are: AF BD EG

the Capital is: D
the partners are: AB CE
```

Problem 3 - RoboCar 2

Stan build a robot car that can drive along a straight line and that can modify its speed every second: However, it can only change its speed up or down one meter per second: If it is going 5 meters per second one second, it can change its speed to 4, 5 or 6 meters per second the following second. Depending on the terrain the car is traveling, Stan can preset its top speed and the total distance the car must travel.

Speed here is taken to mean: Average speed. To say that for example, the speed at a given time is 7 meters per second, this means, that during the given second, the car travels exactly 7 meters.

The car will always travel as fast as possible, given that it must start from standstill and can't crash through the finish line, which might possibly be a blank wall. During the first second, the car's speed is therefore 1 meter per second, and during the last second, it must also go 1 meter per second.

If the car has been given the instruction to go 100 meters at a top speed of 8 meters per second, it will take 22 seconds: traveling during each second for example the following number of meters:

elapsed seconds	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
speed per second		1	2	3	4	5	6	7	8	8	7	7	7	7	7	6	5	4	3	2	1
meters traveled	0	1	3	6	10	15	21	28	36	44	51	58	65	72	79	85	90	94	97	99	100

Read the chart as follows: The speed between second 0 and second 1 is 1 mps and at the end of second 1 meter 1 has been reached. Between second 15 and second 16 the speed is 5 mps and at exactly the 16th second, it crosses the 90th meter line. In all examples, the individual speeds at each second may vary, but the shortest traveling time is always the same.

Stan has added the ability to sense one speed bump on the road. By giving the distance from the starting line of the speed bump, and the speed limit it must obey crossing the speed bump, the car can adjust its speed to accommodate the extra hurdle. During the critical second that it crosses the obstruction, it will be going at or below the speed limit. if the speed bump occurs between one second interval and the next, the car will obey that speed limit during both seconds.
If in the example above, the speed bump occurs at a distance of 50 meters from the start, and the speed limit is 2 on the speed bump, the race will take 25 seconds, and the speeds during each of these 25 seconds could be:

elapsed seconds	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
speed per second		1	2	3	4	5	6	6	6	5	4	4	3	2	3	4	5	6	6	6	5	4	4	3	2	1
meters traveled	0	1	3	6	10	15	21	27	33	38	42	46	49	51	54	58	63	69	75	81	86	90	94	97	99	100

Note here, that between second 12 and second 13 the speed is 2 mps, and so meter 50 is reached and passed in that interval, (at second 12.5) and at that speed.

DATA21 (DATA22 on the second try, will contain 20 lines each containing one integer value.
They represent 5 sets of 4 data:
(a) the distance to be traveled (a number between 10 and 500
(b) the overall speed limit (a number between 2 and 500)
(c) the distance between the speed bump and the starting position
(d) the speed limit over the speed bump

It is your task to write a program that will calculate the shortest time it will take to finish the race for each of the 5 sets of data.

Sample input:

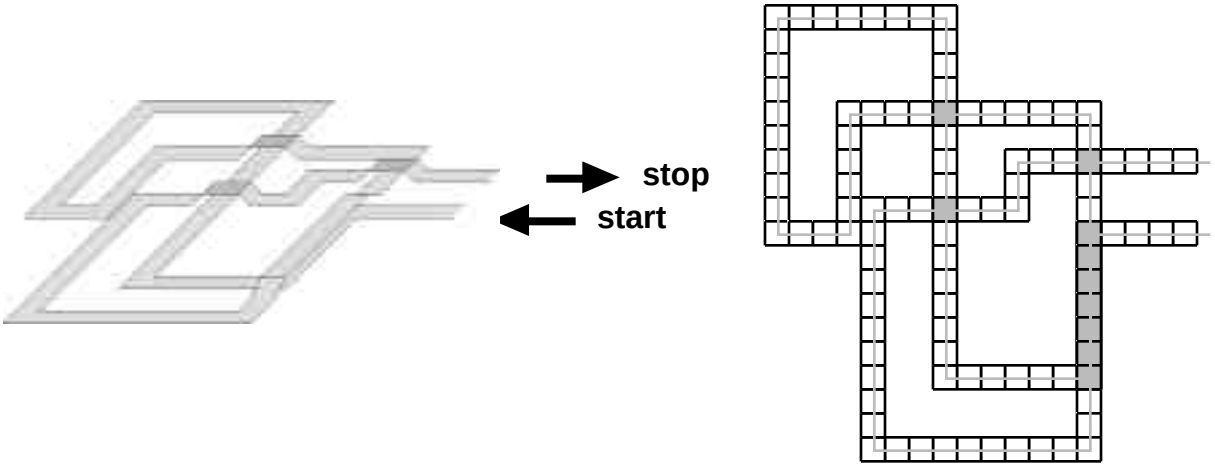
```
100
6
50
2
500
10
30
4
43
2
10
1
250
250
90
16
320
150
310
2
```

Sample output:

```
Stan's car travels the 100 meters in 25 seconds
Stan's car travels the 500 meters in 62 seconds
Stan's car travels the 43 meters in 24 seconds
Stan's car travels the 250 meters in 31 seconds
Stan's car travels the 320 meters in 38 seconds
```

Problem 4 - Daedalus' Garden Path

Daedalus, the designer of the Labyrinth for King Minos of Crete also build a meandering garden path for the King. Of course, the road was paved with gold, and it was expensive. The path was one cubit wide and its length was measured in cubits also: A cubit of length cost the king 5 talents. Where ever the path crossed itself, a bridge was build, at a cost of 100 talents a piece. Occasionally the path ran over itself, creating two levels. Each cubit of the second or succeeding level cost the king an extra 100 talents as well.



Daedalus had his own special code for defining the path. He used integers to indicate straight ways in number of cubits, and the letters L (for "turn left") or R (for "turn right"). The diagram above, for example is based on the following string:
5L6R6R15L7L9L3L5R10R14R9R10R6L2R7

The path is 5+6+6+15+7+9+...+7=114 cubits which normally cost 5x114 = 570 talents
There are three bridges which cost an extra 300 talents and 7 cubits are covered by a second level road (the rise and descend for bridges and elevated pathways are not counted) for an extra 700 talents.
The total cost therefore is 1570 talents.

Daedalus got 5 sets of paths, which can be found in DATA41 (DATA42 for the second try). The file contains 5 lines each containing a string of data no longer than 255 characters. They contain valid positive integers between 1 and 99, separated by capital L or capital R. The string contains no other characters, and starts with a number and ends with a number.
The total length of the path is never longer that 1000 cubits.

Write a program that finds the cost of building each pathway, printing the results as in the examples below.

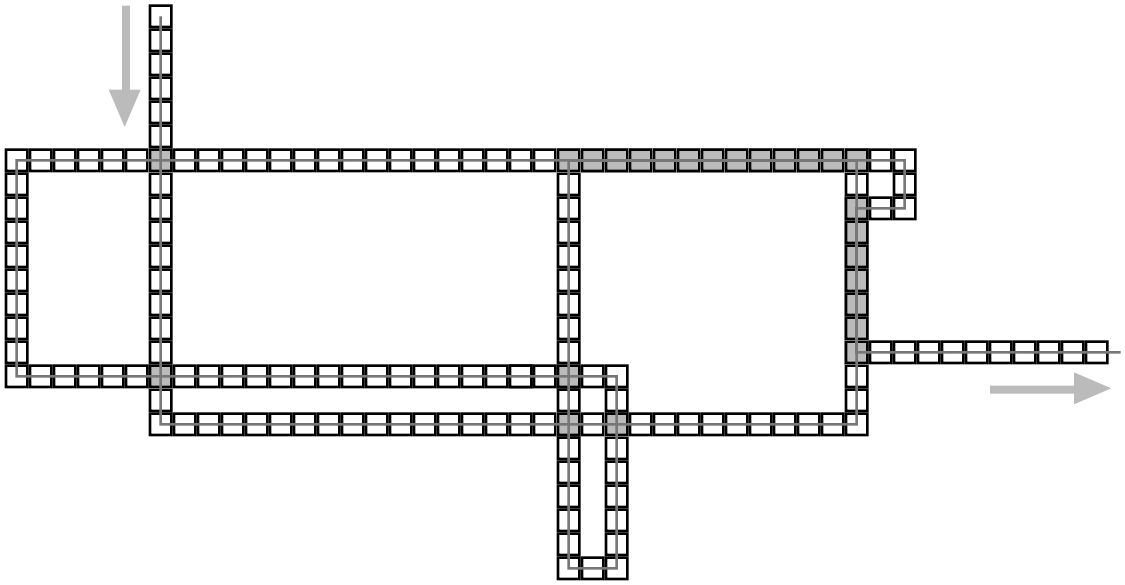
Sample input:

```
5L6R6R15L7L9L3L5R10R14R9R10R6L2R7
15R23R18R25R20R23R18R14R3L11R11R6R8R8R3R16R6R25
99L99R99L1
89
18L29L11L35L9L25R8R2R17R14R2R2L6L10
```

Sample output:

```
the path is 114 cubits long and costs 1570 talents
the path is 253 cubits long and costs 3665 talents
the path is 298 cubits long and costs 1490 talents
the path is 89 cubits long and costs 445 talents
the path is 188 cubits long and costs 3440 talents
```

Note: here is a picture of the last example:



ECOO 2003

Nineteenth Annual Programming Contest

Finals

May 7, 2003

Instructions to Programmers

1. You may use one computer and no printer. You may have the use of a calculator.

2. You may use any programming language, however it must be able to read a text file, and the output must be on the screen. You may use no other software or special libraries.

3. You may use one resource book, one that accompanies the programming software, but no third party manual is allowed.

4. Set up your machine before 11:00 a.m.
   The contest will take place between 11:00 a.m. and 2:00 p.m.

5. You have no more that 2 chances to have your program tested. Input data for the first try is DATAx1 and input data for the second try is DATAx2, where x = 1,2,3 or 4 the problem number.

6. Keep the score sheet near your machine.
   When you are ready to have your program tested, this is the procedure you should follow:
   - Wave the colour coded sheet in the air, to signal the appropriate judge that your program is ready to be tested.
   - The judge will bring over the disk. While two judges are present, copy the file onto your hard drive and RETURN the disk
   - YOU run the program: The judges are there to observe.
   - They will tally your score and you move on the next program. They will take your sheet with them to the Scoring station. It will be returned after it is processed.
   - Once the two judges move away from your machine, their decision, right or wrong, will be final. While deliberating, they may call over a 3rd person to arbitrate: If two of the three agree, that decision is final.

7. At 2:00 p.m. the contest ends, and many teams will probably wish to take the opportunity to have their program judged for the last time. Therefore DURING THE LAST FIVE MINUTES OF THE CONTEST, only step 1 of the judging process will take place, namely the jotting down of the time. The other steps can take place at leisure after 2:00. Therefore, at 2:00 pm stand well away from your machine, to clearly indicate, that you are no longer working on it.

Problem 1 - Power Cipher

Ryan discovered that he could rearrange the first 10 natural numbers by creating a sequence as follows: starting with 1 as the first number in the sequence, he multiplied each number in the sequence by 8. If the number becomes larger than 10 he divides by 11 and takes the remainder. He found, that the first 10 numbers in the sequence: t_1=1, t_2=8, t_3=9, t_4=6, t_5=4, t_6=10, t_7=3, t_8=2, t_9=5, t_{10}=7 were simply a rearrangement of the numbers 1, 2, 3, 4, ..., 10. It does not matter, if he were to start with 1 or any other number between 1 and 10.

The sequence is called the powers of 8 mod 11, for the numbers correspond to:
8^0 mod 11, 8^1 mod 11, 8^2 mod 11, ... , 8^9 mod 11.

Ryan decided to encrypt his messages by taking letter number t_x and placing it in the x^{th} position.

Consider the following message:

```
I'll tell thee what, prince;^
a college of witcrackers cannot flout me out of my humour.^
Dost thou think I care for a satire or an epigram?^
```

The message contains 139 characters (The end of line is marked by the character ^). So Ryan divided the characters in sets of 10, and padded the end with the start of the message, to complete the last set of 10. (The "/" is inserted here for ease of viewing only, it is not part of the encoding)

```
I'll tell /thee what,/ prince;^a/ college o/f witcrack/
ers cannot/ flout me /out of my /humour.^Do/st thou th/
ink I care/ for a sat/ire or an /epigram?^I/
```

Rearranging the characters, the message becomes:

```
Illtl l' e/tatwe,eh h/ ;^ciarpne/ e elooclg/faccikw tr/
enoa tsrcn/ meto lfu /omyf  tuo /h^Droomuu./s toth thu/
iar  eknIc/ saartof  /ianr  ero /e?^agIiprm/
```

Ryan found that powers of 8 could in the same way rearrange numbers mod other primes. In fact, some powers of base different from 8 had that same property.
He could encrypt messages, by sending a 3-number key: (1) the prime modulus, in this case 11, (2) the starting number, in this case 1, and (3) the base for the powers, in this case 8.

The last message was decoded using 11-1-8. However, when using 53-7-20 Using the same message:

```
I'll tell thee what, prince;^a college of witcracker/
s cannot flout me out of my humour.^Dost thou think /
I care for a satire or an epigram?^I'll tell thee wh/
```

it becomes:

```
elicle fnre geeate ltlI,^kctl ccwh;ahpw'o r ea troil/
orooayf  o^mDkuhnt .ltsuhn o timho ue m tntso tu ufc/
 ?laaertn Ir'w eesa^ fI i teooe lapgirt  rhllre hmac/
```

You must write a program that will unscramble Ryan's five messages. The input file, DATA11 (DATA12 for the second try) will contain 4 numbers, each on a separate line, followed by several lines of text. The numbers represent respectively: the modulus, the initial value, the base of powers, and the number of lines of text that follow.
Note that:

1- The modulus is always a two-digit prime number.
2- The repeated (if any) characters at the end of the message do not contain the ^ character, and so the last ^ character in the decoded message spells the end the original message.
3- You may assume that the text contains an exact multiple of (modulus-1) characters.
4- Space characters may occur at the start, the middle or the end of lines, however, in the following sample input, NO space characters occur at the end of lines.

Sample Input (3 of 5 sets only, text from Shakespeare's Much Ado About Nothing)

```
11
1
8
3
Illtl l' etatwe,eh h ;^ciarpne e elooclgfaccikw trenoa tsr
cn meto lfu omyf  tuo h^Droomuu.s toth thuiar  eknIc saart
of  ianr  ero e?^agIiprm
53
15
3
5
iswbl^lware  etb sa hh'ellnaab naof N; n t,wae  miiidcm   oo b dn tb
af neIiosIeiuenhrrtma nhi^,ns hag.ioryntro,ophspnt ia  o n
alyug lIpmius^prthi tr  ewoknolriir   ftneea son  etha
doatyahw; tcethgasdrtn  en^amths,a fiao^s twta   enhet oaoraivf
feeuI vg irl tad dutino ^isN.lnagi   ssc,yoys hag   trm d icdniinahmn
83
12
5
2
lhy!.!as lppeeyleoouwIhP.m^itoe  p c  tsawrPtut  I acyoew Pe
milrouuoi otIlcsol  ! l
```

Sample output:

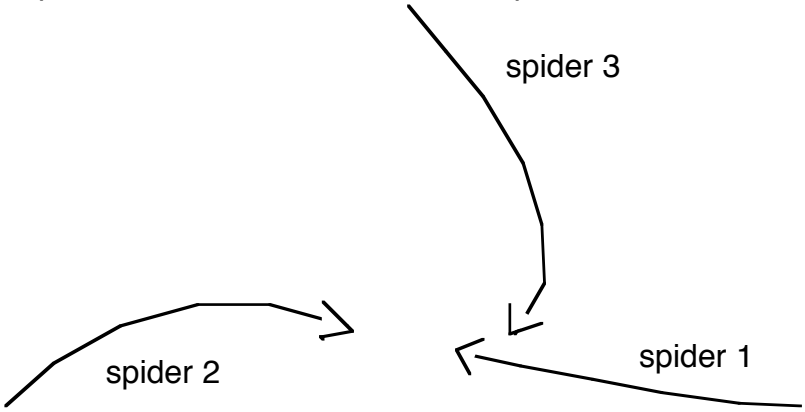
```
I'll tell thee what, prince;
a college of witcrackers cannot flout me out of my humour.
Dost thou think I care for a satire or an epigram?

No; if a man will be beaten with brains,
a' shall wear nothing handsome about him.
In brief, since I do purpose to marry,
I will think nothing to any purpose that the world can say against it;
and therefore never flout at me for what I have said against it,
for man is a giddy thing, and this is my conclusion.

Peace! I will stop your mouth.
```

Problem 2- Chasing Spiders

Consider three spiders chasing each other: Spider 1 chases after spider 2. Spider 2 chases after spider 3 and spider 3 chases after spider 1. They each start out in different positions on a cartesian grid, where both the x-axis and y-axis has been divided into discrete centimeters. They run at different speeds, measured in centimeters per second.



Depending on their positions and speed, one spider will make a catch before the other two. Each spider runs in a predictable way: It looks at its quarry at the start of each second. It then runs in a straight line in that direction for exactly one second, even though during that second its quarry is also moving. At the start of the next second, they look again, adjust their direction and run again for one second. Each spider repeats the procedure until one of them catches another.

If the distance between a spider and its quarry is less than the distance it can crawl in one second, it jumps, which is instantaneous, and it catches its victim. If there is a tie, spider 1 wins over spider 2 or 3, and spider 2 wins over spider 3.

DATA21 (DATA22 for the second try) contains five sets of data: Each set takes up 9 lines, three positive integers for each spider: Its x-position, its y-position and its speed.
Write a program that will read the file and print out which spider will make a catch and how many seconds it will take. All numbers are integers between 2 and 2000

Sample Input: (for convenience each set is in a separate column, however, in the input file, each number occupies one separate line:

5	350	775	150	600
0	1225	950	850	575
8	8	6	10	10
20	100	150	850	525
0	1100	250	800	1225
8	8	15	10	9
105	775	50	275	50
0	825	1000	675	200
8	12	13	9	12

Sample output:

```
spider #2 made the catch after 5 seconds

spider #2 made the catch after 37 seconds

spider #3 made the catch after 43 seconds

spider #1 made the catch after 35 seconds

spider #2 made the catch after 55 seconds
```

Problem 3 - Perimeter

Imagine shapes made from unit squares as in the following examples:

By manually counting the lengths of the sides of the figures, you can verify that their circumferences are in order from left to right: 26, 10, 12, 36 and 20

It is your task to create a program that will find their circumference, given the format as in the sample input below.

The text file DATA31 (DATA32 for the second try) will contain one rectangle filled with minus signs (-). Within the rectangle you will find 5 shapes, each one formed from the first five letters of the alphabet.

please note that:
1- Each letter of a shape is connected to the rest of the shape with at least one of its four sides.
2- Each shape is uniquely determined by the capital letter with which the shape is formed.
3- Each shape is separated from every other shape: They do not touch, not every at the diagonal.

The first line of the input file contains a number between 10 and 80 representing the length of the rectangle (the number of lines of text to be read in) and the second line contains a number between 10 and 80 containing the width of the rectangle (the length of each line)

Sample Input:

```
13
29
----------------------------
---AAAAAAA-------CCCC--------
------AAAAAA--------C--------
--------AAAA-----B-----------
---------AA------B-----------
-----------------B-----------
-----DDDD--DDDD--B-----------
-------D--DD------------------
-------DDDDD--------EEEEE----
---------D---------EEEEEE----
---------D---------EEEEEE----
--------------------EEEEEE----
----------------------------
```

Sample Output:

```
A has a perimeter of 26
B has a perimeter of 10
C has a perimeter of 12
D has a perimeter of 36
E has a perimeter of 20
```

4- Wine Merchant

Three bottles of various sizes are used for measuring wine. The first bottle is completely filled with wine, and the others are completely empty. When a customer comes to buy wine, the merchant uses the three bottles in combination to measure out the correct amount.

For example: The merchant has the three bottles measuring respectively 5 litres, 4 litres, and 2 litres. The 5 litre bottle is completely filled, and the 4 and 2 litre bottles are empty. Now the customer wants exactly 3 litres.

Solution: step 1: pour 2 litres from the first bottle into the third bottle.
step 2: present the first bottle to the customer, which now contains 3 litres

DATA41 (DATA42 for the second try) contains five sets of 4 numbers, each number on its own line. The first three numbers represent the three sizes of bottles #1, #2 and #3. The fourth number represents the amount of wine the customer wants to buy. Bottle #1 is always completely full and bottles #2 and #3 are always empty.

Write a program that will state in the fewest steps possible, how this might be done as in the sample solution below. Any solution that takes more than 7 steps (lines) is considered impossible.

Sample input:

```
19
7
20
14
20
6
20
2
10
18
13
2
20
19
5
1
16
2
12
6
```

Sample output:

```
pour bottle #1 into bottle #2 to get:  12   7   0
pour bottle #2 into bottle #3 to get:  12   0   7
pour bottle #1 into bottle #2 to get:   5   7   7
pour bottle #2 into bottle #3 to get:   5   0 14
present bottle #3
press any key to continue..


pour bottle #1 into bottle #2 to get:  14   6   0
pour bottle #2 into bottle #3 to get:  14   0   6
pour bottle #1 into bottle #2 to get:   8   6   6
pour bottle #2 into bottle #3 to get:   8   0 12
pour bottle #1 into bottle #2 to get:   2   6 12
present bottle #1
press any key to continue..


the problem is impossible to solve
press any key to continue..


pour bottle #1 into bottle #2 to get:   1 19   0
present bottle #1
press any key to continue..


pour bottle #1 into bottle #3 to get:   4   0 12
pour bottle #3 into bottle #2 to get:   4   2 10
pour bottle #2 into bottle #1 to get:   6   0 10
present bottle #1
press any key to continue..
```