A+ Computer Science
Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	Scale
Problem 2	Playtime
Problem 3	Floor Cleaner
Problem 4	Feed Store
Problem 5	Reverse
Problem 6	Spiral
Problem 7	Demo Clean Up
Problem 8	Rook
Problem 9	Word Mix
Problem 10	Rain Boots
Problem 11	Triangle Height
Problem 12	Best Seller

1. Scale

Program Name: scale.java **Input File: scale.dat**

Write a program that takes in an array of integers and produces a new array of scaled integer values. The value of each location will be the original value multiplied by its neighbors.

Input

There will be an unknown number of inputs each on its own line. Each input will contain an unknown number of values separated by spaces.

Output

Display the result of each input on its own line, with each value separated by a single space.

Example Input File

```
1 1 1 1 1
2 3 1 4 5 3
7 4 18 5 2 6
```

Example Output to Screen

```
1 1 1 1 1
6 6 12 20 60 15
28 504 360 180 60 12
```

2. Playtime

Program Name: playtime.java Input File: playtime.dat

All your games track your time played in minutes. You have decided to create a program that will convert the minutes into years, days, hours and minutes.

Notes:

- **Years will be treated as 365 days.**
- **Playtime will never be 0 minutes**

Input

The first line will contain the number of inputs that follow. Each input will consist of a game title and the number of minutes played, in the following format:

gameName,**minutesPlayed**

Output

For each input display the game title followed by how long it has been played in years, days, hours and minutes. Each unit of time is only represented if it's value is greater than 0.

Result Format:

GameTile - `# year(s) # day(s) # hour(s) # minute(s)`

Example Input File

```
5
2048,800
Halo,1097578
Heroes of the Storm,6736732
Infinity Blade,78
Xcom,43
```

Example Output to Screen

```
2048 - 13 hour(s) 20 minute(s)
Halo - 2 year(s) 32 day(s) 4 hour(s) 58 minute(s)
Heroes of the Storm - 12 year(s) 298 day(s) 6 hour(s) 52
minute(s)
Infinity Blade - 1 hour(s) 18 minute(s)
Xcom - 43 minute(s)
```

3. Floor Cleaner

Program Name: floor_cleaner.java **Input File: floor_cleaner.dat**

You have built a floor cleaning robot and now you are writing an algorithm that will report all the locations of the house it could not clean. The robot is assumed to have a layout of the house that includes where the walls, furniture and flooring are. When it cleans it will log locations where it finds unexpected obstructions.

Key:

- 'W' – Wall
- 'F' – Furniture
- '-' – Flooring
- 'B' – Base

Input

The input will start with a 15 by 15 grid of the house and will be followed by an unknown number of locations where obstructions were found, in the following format:
(**column**,**row**)

Output

Display all the locations that were not able to be cleaned in the following format:
(column,row)

Display each location on its own line. Locations must be displayed in ascending order first by row and they by column.

Example Input File

```
WWWWWWWWWW
W-------BW
WF--FW---W
W---FW--FW
WWWWW---FW
W--WW---WW
W-FW----FW
W-WWWF---W
W--------W
WWWWWWWWWW
(2,1)
(1,8)
```

Example Output to Screen

```
(1,1)
(2,1)
(1,5)
(2,5)
(1,6)
(1,7)
(1,8)
```

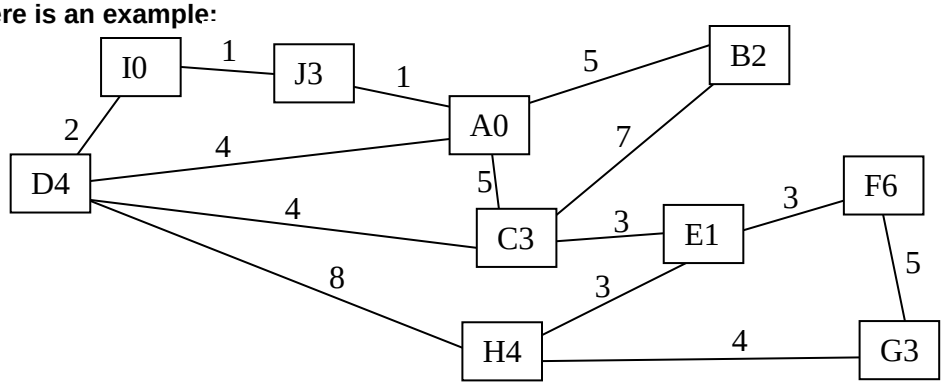
4. Feed Store

Program Name: feed_store.java **Input File: feed_store.dat**

You work at a feed store and want to make your delivery routes more efficient. You have a list of how much feed each nearby farm needs, a map of the roads and the total amount of feed your truck can carry. Write a program that will take in all that data and give you the shortest number of miles for completing your first delivery and the path for the route. A delivery route will always start and end with 'A'. 'A' will always be the store you are delivering from.

Note: There may be several best paths with the same answer, so you may not get the same exact path as the sample data, but your solution needs to be correct and be of the shortest distance.

Here is an example:



Input

The first line will contain a number that indicates how much feed your truck can carry.

The second line will contain how much feed each farm wants. A farm name will always be a single uppercase letter. Each farm will list how much feed they want as 0 to 9 units of feed. The data for each farm will be a single letter followed immediately by a single digit. The data for each farm will be separated by a single comma.

Finally, there will be an unknown number of lines defining the connections between farms and the store / other farms. Connections will be in the following format:

locationA/locationB-distanceApart

Output

Output the distance of the shortest solution and the path for that solution. The format will for the solution will be:

(distance) - path

Example Input File

```
10
B2,C3,D4,E1,F6,G3,H4,I0,J3
A/C-5
A/B-5
A/J-1
A/D-4
I/J-1
I/D-2
C/B-7
C/D-4
C/E-3
H/D-8
H/E-3
H/G-4
E/F-3
F/G-5
```

Example Output to Screen

```
(13) - ACDIJA
```

5. Reverse

Program Name: reverse.java **Input File: reverse.dat**

At a contest you have been asked to write a program that reads in a line of text and reverses the position of all words that begin with vowels. Words that begin with consonants will keep their position in the line of text. You are not sure why you would ever need this, but hey it's worth points.

Input

There will be an unknown number of lines to process and each line will only contain words that are each separated with a single space.

Output

For each input, display the converted line on its own line.

Example Input File

```
apple
Eat apples
snakes are coming into Your house
Axe hat bat other toy octagon me I love oranges
```

Example Output to Screen

```
apple
apples Eat
snakes into coming are Your house
oranges hat bat I toy octagon me other love Axe
```

6. Spiral

Program Name: spiral.java **Input File: spiral.dat**

For an art project you have decided to create a program that centers a spiral pattern in a square character grid.

Notes:

- The spiral will be denoted by '*'
- The spiral will always start in the middle of the grid.
- The movement cycle will be:
  - North East
  - East
  - South East
  - South
  - South West
  - West
  - North West
  - North
- The pattern will change direction if the either of the next two options in the movement cycle are viable without being adjacent to any '*' besides the one that was just placed.
  - When both new directions are valid, the direction that is two away is chosen
    - Example: You are traveling West and both North West and North are valid movements. North would be chosen.
- The pattern will stop once every possible direction is off grid or violates the adjacency rule.

Input

The input will be a single number, representing the size of the grid for the spiral.

Output

Display the spiral where '*' denotes the spiral and '-' denotes area that is not part of the spiral.

Example Input File

```
13
```

Example Output to Screen

```
---*---------
--*--******---
-*--*-----*--
*--*--***--*-
*-*--*---*--*
*-*-*--*--*-*
*-*-*-*-*-*-*
*-*-*---*-*-*
*-*--***--*-*
*--*-----*--*
-*--******--*-
--*-------*--
---********---
```

7. Demo Clean Up

Program Name: demo_clean_up.java **Input File: demo_clean_up.dat**

A demolitions company has hired you to write a program. After every project they have to move away all the debris and they want to know how many truck loads that it will take. Write a program that given the amount of weight the company truck can hold and how much every piece of debris weighs, it will find the minimum number of truck loads to clear the site of debris.

Input

There will be an unknown number of inputs, each on its own line. Each input will start with how much the truck can transport in a single load, followed by the weight of each piece of debris at the site. Each value will be separated by a single comma.

Output

Display the minimum number of loads needed to clear the site for each input. Each result should be on its own line.

Example Input File

```
100,75,75,50
50,25,26,12,12,18
17,2,12,15,16,8,8,8
```

Example Output to Screen

```
3
2
5
```

8. Rook

Program Name: rook.java **Input File: none**

You have just learned how to output text to the screen and your teacher has challenged you to create an ascii art of a chess piece. You have decided to make your favorite piece, the rook.

Input
None.

Output
The rook art, exactly as shown below. There is an extra row of numbers added for your convenience, but the row of numbers should **NOT** be included in your output.

Output to Screen

```
  ___   ___   ___
  | |__| |__| |
  |           |
   \_________/
    \_______/
     |     |
     |     |
     |     |
     |     |
     |_____|
  __/       \__
 /             \
/_______________\
01234567890123456
```

9. Word Mix

Program Name: word_mix.java **Input File: word_mix.dat**

Write a program that will combine two words into a new one. The length of the two words will be unknown. The new word will be the same length as the shorter word. The letters at even indexes will be taken from the first word at the corresponding indexes and the letters at odd indexes will be taken from the second word at the corresponding indexes.

Input
The input will contain two words on a single line, separated by a single space.

Output
Display the mixed word.

Example Input File
```
stat better
```

Example Output to Screen
```
seat
```

10. Rain Boots

Program Name: rain_boots.java **Input File: rain_boots.dat**

You love your new rain boots and want to keep them as clean as possible. Using satellite data you have gridded the area nearby as either muddy or not muddy. Write a program that will determine the fewest number of muddy puddles you must walk through to reach a destination.

Key:

- 'M' – Muddy Puddle
- '-' – No Mud
- 'S' – Start (No Mud)
- 'E' – End (No Mud)

Input

There will be an unknown number of inputs. Each input will contain an 8 by 8 grid that represents the area nearby. There will be a single line containing a single '-' separating each input.

Output

For each input, display the fewest number of muddy puddles you must step through to reach the destination.

Example Input File

```
S-------
-M------
--M-----
---M----
----M-MM
-----M--
-----M--
-----ME-
-
SM------
MM------
--------
MMMMMMMM
--------
MMM-----
MEM-----
-----ME-
-
-------M
-MMMMM-M
-MS--M-M
-MMM-M-M
-----M-M
MMMMMM-M
-------M
EMMMMMMM
-
-M-MEM-M-
M-M-M-M-M
-M-M-M-M-
M-M-M-M-M
-M-M-M-M-
M-M-M-M-M
SM-M-M-M-
M-M-M-M-M
```

Example Output to Screen

```
1
2
0
5
```

11. Triangle Height

Program Name: triangle_height.java **Input File: triangle_height.dat**

Your Math teacher asked you to write a program to help him generate homework solutions. The program will need to find the height of a triangle given its area and its base length.

Formula:

a = (h*b)/2
(a – area, b – base length, h – height)

Input

The first line will contain a single integer `n` that indicates the number of lines that follow. Each line will include the area and base length of a triangle, in the following format:

area,**base**

Output

For each input display the height of the given triangle with 2 decimal places, in the following format:

The height of the triangle is #.## units

Example Input File

```
4
200.533,40.5
10.6,1.11
30,30
3333,50.7
```

Example Output to Screen

```
The height of the triangle is 9.90 units
The height of the triangle is 19.10 units
The height of the triangle is 2.00 units
The height of the triangle is 131.48 units
```

12. Best Seller

Program Name: best_seller.java **Input File: best_seller.dat**

You run a store and want to know where you make the most money and which items sell the most. You have a list of your items, how many times each has sold and how much profit you make each sale. The total profit on an item is the number of sales times the profit of the item. Write a program that will take in this data and produce a sorted list using the following rules:

Sorting Rules:

- Sort items in descending order by total profit
- For items that have the same total profit, sort in descending order by number of sales
- For items that have the same total profit and number of sales, sort in ascending order by name

Input

There will be an unknown number of inputs, with each on its own line. The format for each input will be:

itemName,**numberOfSales**,$**profitEachSale**

Output

Display the sorted items list, with each item on its own line. The format for displaying the item will be:

$**totalProfit** – **itemName**/**numberOfSales**

Example Input File

```
pens,120,$.12
binders,120,$1.15
hats,24,$6.00
shoes,12,$12.00
rings,44,$22.86
coats,14,$28.64
boots,12,$12.00
```

Example Output to Screen

```
$1005.84 - rings/44
$400.96 - coats/14
$144.00 - hats/24
$144.00 - boots/12
$144.00 - shoes/12
$138.00 - binders/120
$14.40 - pens/120
```