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	Rocket
Problem 2	Ball Drop
Problem 3	Balanced Words
Problem 4	Hiking Challenge
Problem 5	Final Grade
Problem 6	Clearing the Path
Problem 7	Cliff Climber
Problem 8	Rounded Adder
Problem 9	Birthday Finder
Problem 10	Unique Paths
Problem 11	Synonym Placer
Problem 12	Shape Rotation

For more Computer Science practice tests and materials, go to www.apluscompsci.com

1. Rocket

Program Name: Rocket.java

You are making a text based space game and you want it display a rocket with a launch pad when the game loads. Write a program that displays the below acsii art to the screen.

```
|--|
|\/|
|/\|        /\
|--|       /__\
|\/|______|    |
|/\|xxxxxx|____|
|--|      |N   |
|\/|      |A   |
|/\|      |S   |
|--|      |A   |
|\/|      |    |
|/\|      |____|
|--|      /\  /\
|\/|     |  ||  |
|/\|     |  ||  |
|--|-----------------
|\/|\/|\/|\/|\/|\/|\/|
|/\|\/|\/|\/|\/|\/|\/|
```

Input

None.

Output

Displays the rocket and launch pad.

Example Output to Screen

```
|--|
|\/|
|/\|        /\
|--|       /__\
|\/|______|     |
|/\|xxxxxx|_____|
|--|      |N    |
|\/|      |A    |
|/\|      |S    |
|--|      |A    |
|\/|      |     |
|/\|      |_____|
|--|      /\  /\
|\/|     |  ||  |
|/\|     |  ||  |
|--|-----------------
|\/|\/|\/|\/|\/|\/|\/|
|/\|\/|\/|\/|\/|\/|\/|
```

2. Ball Drop

Program Name: BallDrop.java Input File: ball_drop.dat

Your Physics teacher asked you to write a program that calculates how long it would take a ball to reach the ground when dropped from a given height, at an initial velocity of 0 meters per second.

The formal is:

time = (2*distance/ gravity)$^{1/2}$

Gravity = 9.8 m/s^2

Input

There will be 4 inputs, each on their own line. Each input will be a single number representing the drop height in meters.

Output

For each input display the number of seconds it would take for the ball to hit the ground, rounded to 2 decimal places.

The format of your output should be:

x.xx second(s)

Example Input File

```
548.2
172
51
9.8
```

Example Output to Screen

```
10.58 second(s)
5.92 second(s)
3.23 second(s)
1.41 second(s)
```

3. Balanced Words

Program Name: BalancedWords.java Input File: balanced_words.dat

You have been asked to write a program that will count the number of balanced words in a text file. Words of odd length with odd asci totals are balanced and words of even length with even asci totals are balanced.

Input

There will be an unknown number of text lines as input. The text will not have any punctuation.

Output

The format for your output will be:

The provided text has xx balanced word(s).

Example Input File

```
This text is very random and has no point
It should be used for testing purposes only
I hope this problem goes well for your team
```

Example Output to Screen

```
The provided text has 13 balanced word(s).
```

4. Hiking Challenge

Program Name: HikingChallenge.java **Input File: hiking_challenge.dat**

You and your friends love hiking and enjoying covering ground quickly. You have decided it would be fun to have a challenge time for your hikes. You want to build a program that will generate tough, but achievable hiking time challenges.

You have broken many of your hiking maps into cells and assigned each cell as trail, wooded, overgrown or steep.

You have estimated the average time it should take to travel though each terrain as:

Terrain Type	Minutes to travel
Trail	3
Wooded	8
Overgrown	12
Steep	20

Now you need to build a program that analyzes terrain maps and gives you a challenge time for different hiking sites. You have decided your challenge times should be based on the fastest available path, not the shortest.

Input

The input file will consist of 3 inputs.

The first line of each input will contain your starting location and the destination, in this format:

{(startingColumn, startingRow), (endColumn, endRow)}

All the maps will be 5 by 5.

The key for the map is:

Terrain Type	Symbol
Trail	T
Wooded	W
Overgrown	O
Steep	S

Output

For each input display the goal time:

The format for your output will need to be:

Your time to beast is xxxx minutes.

Example Input File

```
{(0,0),(3,3)}
TTTTT
OSSSW
STOST
SSSSS
SWOWS
{(2,0),(2,3)}
TTTOO
TSSWO
TTOSO
TSSSO
TTTSS
{(0,1),(2,2)}
TTWTT
OSSTT
WSSTT
SSSTT
TTTTT
```

Example Output to Screen

```
Your time to beat is 46 minutes.
Your time to beat is 27 minutes.
Your time to beat is 35 minutes.
```

5. Final Grade

Program Name: FinalGrade.java **Input File: final_grade.dat**

Your teachers have had enough… They will no longer tell you what you need to make on your final to get an A in a course. The math is fairly simple and you have decided to write a program to calculate the final grade needed to receive an A in any class.

Your teachers do not round semesters grades, so your semester grade must be 90.0 or higher to get an A. Your school has 3 grading periods per semester and one final. The grade you receive in a class is 25% of your final score plus 75% of the average of your 3 grading periods.

In equation form it looks like this:

SemesterGrade = GradingPeriodAverage*.75 + FinalGrade*.25

You release that you must first solve this equation before writing you program!!

Input

The program will contain four inputs, each on their own line.

Each line of input will be the following format:

1st_Grading_Period, 2nd_Grading_Period, 3rd_Grading_Period

Output

Display the grade needed on the final to get an A, rounded to 2 decimal places.

Example Input File

```
87.7, 48.23, 90.54
83.7, 94.2, 93.1
77.23, 98.65, 92.0
88.77, 87.44, 85.11
```

Example Output to Screen

```
133.53
89.00
92.12
98.68
```

6. Clearing the Path

Program Name: ClearingThePath.java **Input File: clearing_the_path.dat**

You are in the process of building a game where the player tries to clear a path through rubble, using explosives. A single explosive can remove 1 rubble square in a level. Each level in the game will use a different rubble map and the number of explosives available to the player will vary map to map.

You want your game to have a lot of levels, so you created an algorithm for generating levels. You have found that some of your generated maps were unsolvable. To resolve this issues, you must right a second algorithm to verify that map is beatable with a given number of explosives.

Input

The input will include an unknow number levels.

Each level will be the following format:

Bombs:MapWidth**x**MapHeight
MapGrid

The key for levels will be:

Terrain Type	Symbol
Start	S
End	E
Open	O
Rubble	R

Output

For each input display: Keep or Delete

- Display Keep when the map is solvable
- Display Delete when the map is not solvable

Example Input File

```
5:7x1
SOROROE
3:7x1
SORRRRE
2:4x4
RRRR
RRER
RRRS
RRRR
1:4x4
ROOR
RRRR
RRRS
RERR
```

Example Output to Screen

```
Keep
Delete
Keep
Delete
```

7. Cliff Climber

Program Name: CliffClimber.java **Input File: cliff_climber.dat**

You are preparing for a test run of grappling hook you have designed, with a releasable hook. You plan to use your new grappling hook to both climb up and repel down cliffs. Because the hook can be released you will only need one rope. The rope is heavy, so you only want the exact amount needed for your test site.

Each test site has an elevation map and you have decided to create a program that will read in an elevation map and tell you how much rope you should take. Your test run will always start at the top of the left most cliff. Each time there is an elevation change from that point on you will have to use your grappling hook to climb or repel some distance. After each repel/climb your rope is returned.

Input

The program will have a single input with an unknown number of lines. Each line will consist of hyphens that represent emptiness and C’s that represent ground/cliff faces.

Note: The input will always be rectangular.

Output

Display the length of rope needed for the site, in the following format:

```
x units of rope are needed for this site.
```

Example Input File

```
--------------------------------------------------
------C----------C--------------------------------
------C----------C--------------C-----------------
------C----------C--------------C------C----------
------C----------C--------------C------C----------
------C----------CC----C--------C------C----------
C-----C----------CC----C-----C--C--C---C----------
C-----C----C-----CC----C-----C--C--CC--C----------
C-----C----C-----CCCCCCCCC--CC--C--CCC-C---CCCCC--
C-----C----CC---CCCCCCCCCCCCCCCCC-CCCCCC---CCCCCC-
CCCCCCC----CC-C-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-
CCCCCCCC--CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC-
CCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCC-CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Example Output to Screen

```
10 units of rope are needed for this site.
```

8. Rounded Adder

Program Name: RoundedAdder.java Input File: rounded_adder.dat

A company has asked to write a program that adds rounded numbers together. They want the program to read in numbers, round them and then total their value.

Note: The program will produce an incorrect output if the numbers are added before being rounded.

Input

The first line will contain the number of decimal places needed. Then there will be an unknown number of values each on their own line.

Output

Display the total on the rounded number.

Example Input File

```
2
7.5399
17.9355
2.9999
3.1888
.99968
.55777
48.65984
14.55447
78.16936
14.3568
```

Example Output to Screen

```
188.97
```

9. Birthday Finder

Program Name: BirthdayFinder.java **Input File: birthday_finder.dat**

For fun you and your friends have decided to build a program that can determine someone's birthday given a date and their age in days on that date.

Input

The program will contain an unknown number of inputs, each on their own line.

Each input will be in the following format:

mm/dd/yyyy – daysOlds

Output

For each input display its birthdate on its own line:

Each input will be in the following format:

mm/dd/yyyy

Example Input File

```
05/30/2005 - 5
03/03/2005 - 368
7/01/1090 - 1582
```

Example Output to Screen

```
05/25/2005
02/29/2004
03/02/1086
```

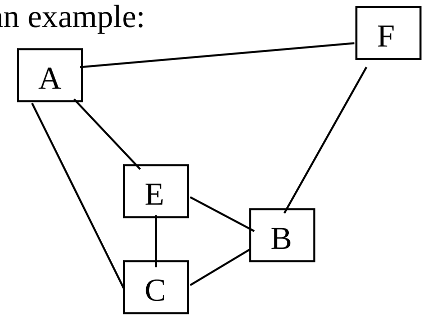
10. Unique Paths

Program Name: UniquePaths.java **Input File: unique_paths.dat**

A road planner has come to you for help. He has a map of cities/roads and he wants you to write a program that will tell him how many unique paths there are from a given city to another. He is not concerned with the distance between the two city, just how many unique paths there are from one city to another.

Two paths of the same length that contain the same cities are considered the same and should only be counted as single path.

Here is an example:



Notes:

- **A path cannot visit the same city more than once.**
- **The sample input listed is for the above map!**

The paths from A to F are: AF, AEBF, AECBF, ACBF, ACEBF.

There would be 4 unique paths, because AECBF and ACEBF visit the same cities and are of equal length.

Input

City names will always be letters ranging from A to Z.

The first line of input will be which cities you are trying to connect.

The format for the first line will be:
startingCity to EndingCity

Next there will be an unknown number of lines listing the cities that are connected.

The format for connected cities:
CityA – CityB

Output

Display the number of unique paths in the following format:
The number of unique paths connecting the two cities is xxx.

Example Input File

```
A to F
A-E
A-C
E-C
C-B
E-B
B-F
A-F
```

Example Output to Screen

```
The number of unique paths connecting the two cities is 4.
```

11. Synonym Placer

Program Name: SynonymPlacer.java Input File: synonym_placer.dat

You keep getting marked off on your papers for over using words, so you have decided to write a program to make your life easier. The program will replace some of a given word with one of its synonyms.

Your program will take in a word, synonyms for that word and a chunk of text. The first time the word is found it will be left alone, the 2nd the word is found it will be replaced with the first synonym, the third time the word is found it will be replaced with the 2nd synonym and so on. After all the synonyms have been used the cycle will start back with original word.

Input

The first line of input will be a word followed by some of it's synonyms, each separated by a comma.

Every line after the first will be text that needs to be processed by your algorithm.

Note: The number of synonyms and lines to be processed are unknown.

Output

Display the new text with the inserted synonyms.

Example Input File

```
good,wonderful,great,excellent
I like to get good grades. My parents like when I get good grades
and then I get to do more good activities. My favorite good activity
is playing on the computer. I am very good at computer games. I am
so good that I am ranked gold on LoL. Believe me, when I say that is good.
```

Example Output to Screen

```
I like to get good grades. My parents like when I get wonderful grades
and then I get to do more great activities. My favorite excellent activity
is playing on the computer. I am very good at computer games. I am
so wonderful that I am ranked gold on LoL. Believe me, when I say that is great.
```

12. Shape Rotation

Program Name: ShapeRotation.java Input File: shape_rotation.dat

You are working on building a game that is similar to Tetris and need to write and algorithm for rotating shapes. The bounds of the shapes will always be rectangle and the program will only need to rotate shapes clockwise by 0, 90, 180 or 270 degrees.

Input

There will be an unknown number of inputs. Each input will include dimensions for a shape, the rotation needed and finally the shape.

The first line of each input will define the bounds of the shape and what rotation is needed.

The format for this line will be:
width**x**height – degrees

The shape will be given over multiple lines. *'s will define the shape and –'s will be used for unused space.

Output

Display each rotated shape, separated by blank lines.

Example Input File

```
4x4 - 180
----
-*--
--**
--*-
6x2 - 90
******
--**--
3x3 - 0
---
*--
-*-
```

Example Output to Screen

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

-*
-*
**
**
-*
-*

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