Problem 1

Reversal of Field

[Novice teams only]

2 points

JAVA: program name must be prob01.class
C /C++ program name must be: prob01.exe

Task Description

A simple palindrome is a word, phrase, or number that reads the same both left-to-right and right-to-left (ignoring punctuation). You will write a program to determine if a given input string is a palindrome.

Program Input

You will prompt the user for an input string from the keyboard, and read it.

```
Enter string: Madam, I’m Adam.
```

Program Output

For each input string, you will echo the string back to the console, followed by an indication as to whether or not it is a palindrome. Examples:

```
Madam, I’m Adam.
-- is a palindrome
```

Another input/output example:

```
Enter string: 12345321
12345321
-- is not a palindrome
```

Problem 2

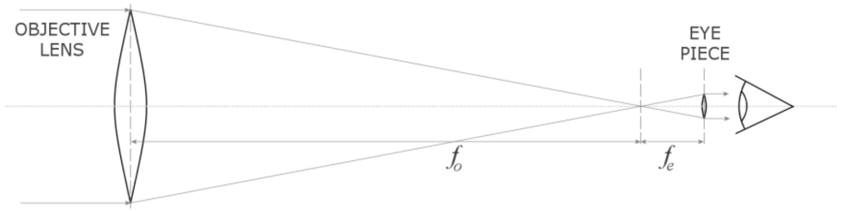
Magnificent Views   [Novice teams only]

3 points

JAVA: program name must be prob02.class
C /C++ program name must be: prob02.exe

Task Description

A telescope is typically composed of two optical elements: the primary or objective, which is usually a large lens or curved mirror, and the eyepiece. The objective is mounted securely in a tube or truss and it is only removed for cleaning or transport. Eyepieces, however, are fitted into a focusing tube and can be easily swapped to provide different levels of magnification. The design of the eyepiece also determines its field of view, which is an angular measure of the width of the view in the eyepiece.



Write a program to compute magnification and true field of view for combinations of objectives and eyepieces. Magnification M is computed by dividing the focal length of the objective f_o by the focal length of the eyepiece f_e. Focal lengths are commonly specified in millimeters. True field of view F_T is computed by dividing the eyepiece's apparent field of view F_A by the magnification M. Field of view is commonly expressed in degrees.

$$M = f_o / f_e$$
$$F_T = F_A / M$$

Program Input

The program will read a series of lines from the keyboard. The three numbers on each line represent, in order, the focal length of the objective, the focal length of the eyepiece, and the apparent field of view of the eyepiece. The program should exit when it reads an input line with three zeros.

```
1000 25 60
1000 7.9 55
750 15 65
500 12 66
1000 20 35
0 0 0
```

Program Output

The program must print the magnification and true field of view for each line of input. Results should be rounded to two places after the decimal. Rounding errors within +/- 0.01 will be accepted. Errors greater than +/- 0.01 will be judged incorrect.

```
40.00 1.50
126.58 0.43
50.00 1.30
41.67 1.58
50.00 0.70
```

Problem 3

Checkerboard

3 points

JAVA: program name must be prob03.class
C /C++ program name must be: prob03.exe

Task Description

You and a friend want to play a game of checkers, but you have no board! Oh, no! Thinking quickly, you reach in your pocket and find eight pennies, and somehow your friend finds eight dimes (don't ask where they came from). Now all you need is a checkerboard. You glance at the computer and printer and think of a great use for that expensive computational power!

Write a program to print a checkerboard pattern.

Program Input

Each line of input will contain a positive integer representing the size of an individual checker square. The largest possible input value is 9. The last line of the input is the number zero.

```
2
1
4
0
```

Program Output

The program must print an 8x8 checkerboard corresponding to each input value. Use the # character for black squares and the . character for white squares. Each checkerboard must be separated by a blank line.

```
##..##..##..##..
##..##..##..##..
..##..##..##..##
..##..##..##..##
##..##..##..##..
##..##..##..##..
..##..##..##..##
..##..##..##..##
##..##..##..##..
##..##..##..##..
..##..##..##..##
..##..##..##..##
##..##..##..##..
##..##..##..##..
..##..##..##..##
..##..##..##..##

#.#.#.#.
.#.#.#.#
#.#.#.#.
.#.#.#.#
#.#.#.#.
.#.#.#.#
#.#.#.#.
.#.#.#.#
```

continued…

```
####....####....####....####....
####....####....####....####....
####....####....####....####....
####....####....####....####....
....####....####....####....####
....####....####....####....####
....####....####....####....####
....####....####....####....####
####....####....####....####....
####....####....####....####....
####....####....####....####....
####....####....####....####....
....####....####....####....####
....####....####....####....####
....####....####....####....####
....####....####....####....####
####....####....####....####....
####....####....####....####....
####....####....####....####....
####....####....####....####....
....####....####....####....####
....####....####....####....####
....####....####....####....####
....####....####....####....####
####....####....####....####....
####....####....####....####....
####....####....####....####....
####....####....####....####....
....####....####....####....####
....####....####....####....####
....####....####....####....####
....####....####....####....####
```

Problem 4

Redundant Acronym Syndrome Syndrome

4 points

JAVA: program name must be prob04.class
C /C++ program name must be: prob04.exe

Task Description

From Wikipedia, the free encyclopedia:

> *The term RAS syndrome refers to the use of one of the words that make up an acronym as well as the abbreviation itself, thus in effect repeating that word. It stands for "Redundant Acronym Syndrome syndrome," and is itself a humorous example of a redundant acronym.*

Write a program that generates RAS syndrome acronyms.

Program Input

Each line of the input contains one or more words. The line of text will be no longer than 80 letters, spaces, and/or punctuation. The last line of the input is the word END.

```
automated teller machine
random access memory
alternating current
scholastic aptitude test
international standard book number
END
```

Program Output

The program must print the acronyms in RAS syndrome format. For this program you can assume that the repeated word is always the last word of the input line. Acronyms must be upper-case, regardless of the case of the input.

```
ATM machine
RAM memory
AC current
SAT test
ISBN number
```

Problem 5

Houston Skyline

5 points

JAVA: program name must be prob05.class
C /C++ program name must be: prob05.exe

Task Description

Write a program that will produce a silhouette of the Houston skyline, given the height and placement of a series of buildings.

Program Input

Prompt the user for the start location (the left edge) [1..60], the width [1..40], and height [1..20] of each building. Note that the dimensions of a building are a measure of its interior, and does not count the outside edges. A building of width 1 and height 6 will actually consume 3 columns and 8 rows on the silhouette. Continue prompting for more buildings until a zero is entered for the start location. The entire skyline will fit within 50 columns wide and a maximum height of 20 rows. There will be no more than 16 buildings.

```
Enter building #1's start:    3
Enter building #1's width:    3
Enter building #1's height:   3

Enter building #2's start:    10
Enter building #2's width:    5
Enter building #2's height:   2

Enter building #3's start:    4
Enter building #3's width:    3
Enter building #3's height:   7

Enter building #4's start:    10
Enter building #4's width:    1
Enter building #4's height:   10

Enter building #5's start:    0
```

Program Output

Output to the screen the outline of the resulting silhouette. Smaller buildings may be completely hidden by larger ones.

```
         +-+
         | |
         | |
   +---+ | |
   |   | | |
   |   | | |
   |   | | |
  ++   | | |
  |    | | +---+
  |    | |     |
  |    | |     |
--+    +-+     +----------------------------------
12345678901234567890123456789012345678901234567890
```

Problem 6

PERLs Before Swine

6 points

JAVA: program name must be prob06.class
C /C++ program name must be: prob06.exe

Task Description

Larry Wall didn't like conditional statements in traditional programming languages, so he invented a new language named PERL, an acronym for Practical Extraction and Report Language. PERL interprets conditional statements in a form that (in Mr. Wall's view) more closely resembles human language. Did we really need another programming language? Apparently Mr. Wall thought so. Life for millions of programmers might be different if instead he had written a little program to invert the conditionals of languages the rest of us already know and love. Then he could have written conditionals as he pleased without the need for a new language that some argue is far more confusing than the ones we already had. Would the world have been a better place? We'll never know.

Write a program to convert a conditional into PERL syntax.

Program Input

The input will consist of single-line conditional statements easily understood by first-year programming students in junior-high schools world wide. PERL, on the other hand, rejects these statements as errors.

```
if( a < min ) min = a;
if( 4*a*c - b*b < 32*c + 9 ) r0 = - b*c - sqrt( 3*a - c*c/17 );
```

Program Output

The program must rearrange each conditional into a form that Mr. Wall felt we should all use. After all, it's more like human language and therefore easier to understand. Isn't it?

```
min = a if( a < min );
r0 = - b*c - sqrt( 3*a - c*c/17 ) if( 4*a*c - b*b < 32*c + 9 );
```

Problem 7

It’s All Greek To Me

6 points

JAVA: program name must be prob07.class
C /C++ program name must be: prob07.exe

Task Description

As computer AI capabilities increase, humans will interact with computers more and more through natural (human) language, both verbal and written. Constructing this technology requires the creation of algorithms that can understand and generate the syntax of natural language.

Write a program to generate contractions of Greek root-suffix pairs. Vowel contractions are formed using the (abbreviated) rules in the table below. For your convenience, Greek letters will be transliterated using the Roman alphabet.

		suffix begins with…						
		a	e	i	y	o	ou	w
root ending…	a	a	a	ai	a	w	w	w
	e	y	ei	ei	y	ou	ou	w
	o	w	ou	oi	w	ou	ou	w

The root word will end in a, e, or o, corresponding to the vowels in the leftmost column. The suffix will begin with a, e, i, y, o, ou, or w, corresponding to the vowels in the topmost row. The resulting contraction is located at the intersection of the two vowels. For example, a+o=w.

Program Input

Each line of the input file contains a root word and suffix separated by a dash. The last line of the file is the word END.

```
katathe-amenos
eggua-omen
marture-omeny
END
```

Program Output

The program must print the contractions of the root-suffix pairs.

```
katathymenos
egguwmen
marturoumeny
```

Problem 8

1337 XL8T0RZ

7 points

JAVA: program name must be prob08.class
C /C++ program name must be: prob08.exe

Task Description

Im in ur c0d3 w4rz, h4xxorz ur src!!!11!! 1337 5p34k iz a kr1pt1k 4nd 3var-ch4ng1ng s14ng 0f cyf0rz & 1nt3nshunel mi5p311ing5 uz3d on t3h 1ntarw3b. 41th0 1337 h4v bc0m3 m0r3 ma1ns7r34m 1n r3c3n7 yrz, itz u54g3 iz r4r3 0u751d3 k0mp00t0r g33k ku17ur3. s0 u th1nk 1337 r0xx0rz & full uv w1n, jus no in rl m05+ ppl d0n k4r3x0rs, u n00b.
Write a program to translate 1337 into English.

Program Input

The input file contains an abridged 1337 to English lexicon and a series of lines of 1337 text.

```
[L3X1K0N]
enuf:enough
ur:you're
u:you
cu:see you
ub3r:super
18r:later
gf:girlfriend
bf:boyfriend
rl:real life
d00d:guy
m4d:mad
r0x0rz:rocks
r00lz:rules
w00t:hooray
pwn:defeat
pwnt:defeated
t3h:the
ph33r:fear
gratz:congratulations
r1p:copy
plz:please
mt:mistell
ty:thank you
re:welcome back
0r1y:oh, really
eet3d:ate
chzbrgr:cheeseburger
h4x:hacks
1337:elite
sk001:school
thx:thanks
np:no problem
meh:my
ppl:people
l4m3r:lamer
[1337]
18r d00d ur so ub3r
w00t meh m4d 1337 sk1llz r00lz
[3ND]
```

Program Output

The program must translate the 1337 text to English. All compares must be case insensitive.

```
later guy you're so super
hooray my mad elite skills rules
```

Problem 9

Mirror to the Stars

7 points

JAVA: program name must be prob09.class
C /C++ program name must be: prob09.exe

Task Description

Astronomers use a wide variety of telescope optics to observe the night sky -- and the sun, for that matter. These telescopes use combinations of mirrors and/or lenses to focus light into the observer's view. Depending on the configuration, the image in view may be inverted (upside-down), reversed (mirror-image), or both. Some telescopes do actually produce an image that is neither reversed nor inverted, but those telescopes do not interest us for this contest. Write a program that can invert and/or reverse an image.

Program Input

The program will read a series of images from an input file (prob09.in). Each image is preceded by a line with the image name, dimensions, and one or two letters indicating which transformation needs to be performed: I for inversion, R for reversal. The maximum image dimensions are 36 wide by 24 high.

```
ORION 13 7 R
.*...........
...........*.
.......*.....
...*..*......
.....*.......
*............
............*
CASSIOPEIA 13 7 IR
...*.........
.............
*............
.........*...
......*......
.............
............*
```

Program Output

The program print each image transformed according to the input transformation letter(s). Each image must be preceded by its label.

```
ORION
...........*.
.*...........
.....*.......
......*..*...
.......*.....
............*
*............
CASSIOPEIA
*............
.............
......*......
...*.........
............*
.............
.........*...
```

Problem 10

uPhone Two-Point Zoom

8 points

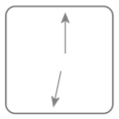
JAVA: program name must be prob10.class
C /C++ program name must be: prob10.exe

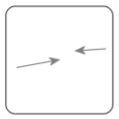
Task Description

Pear Computer Corporation is developing a product called the uPhone. One of the features of the uPhone is the ability to zoom and rotate photos in and out using two-finger pinch and expand. Pear Computer Corporation is convinced people will pay big bucks for this kind of feature. The touch-screen hardware has already been developed, now they just need a little software magic. That's where you come in.
Write a program to convert two-point touch-screen gestures into parameters that control movement, zoom and rotation.

Program Input

Every line of input contains four pairs of positive integers, each pair representing x and y screen coordinates. Each finger touching the screen produces two pairs of x-y coordinates: the start point and the end point, hence, four total pairs. The numbers are: start1-x, start1-y, end1-x, end1-y, start2-x, start2-y, end2-x, and end2-y. Luckily for you, code already exists for filtering invalid gestures, so all input will result in valid operations. The program must exit when it reads a line that contains all zeroes.





```
18 41 96 57 184 123 141 99
58 24 42 36 148 176 164 98
97 81 97 146 81 48 65 16
0 0 0 0 0 0 0 0
```

Program Output

The program must print four numbers for each line of input. The first two numbers represent relative x and y motion of the photo. Relative motion must be computed from the difference of the centers of the end points minus the centers of the start points. The third number represents the zoom factor expressed as a real number. To compute this zoom factor, divide the distance between the end points by the distance between the start points. The distance d between two points $(x1,y1)$ and $(x2,y2)$ is found using the following formula:

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

The fourth number indicates counter-clockwise rotation expressed in radians. For this computation you need to know how to compute the angle between two lines. For two lines with slopes m_1 and m_2, the angle a between the lines is given by the formula:

$$\tan a = \frac{m_2 - m_1}{1 + m_1 m_2}$$

For this program you will assume there are no vertical lines (thus no infinite slope m) and that all rotations are within the range of +/- 90 degrees (1.571 radians). The zoom factor and rotation angle must be accurate within a margin of error of +/- 0.001. Larger errors will be judged as incorrect.

```
17 -4 0.332 0.292
0 -33 0.775 -0.566
-8 17 3.651 0.210
```

* In Java, C, and C++ you'll use the **atan** function to calculate the arctangent. So if tan(a) = b, then a=atan(b). In Pascal this same function is called **ArcTan**.

Problem 11

Diffusion Limited Aggregation

9 points

JAVA: program name must be prob11.class
C /C++ program name must be: prob11.exe

Task Description

Diffusion-limited aggregation (DLA) is a process whereby particles undergoing a random (or simulated random) motion cluster together to form aggregates of such particles. The fractal patterns created by DLAs resemble patterns formed in nature, including river networks and lightning.

Write a program to generate a DLA diagram.

Program Input

The first section of input (read from file prob11.in) is an 11x11 array of previously generated random integers. These numbers could represent, for example, the elevation data for a square section of land. The second section contains x and y coordinates (in that order), which are the starting locations for the particles (such as rain drops) whose paths of motion will form the DLA diagram. The x axis increases left-to-right, and the y axis top-to-bottom. The point (0,0) is in the top-left corner. The input ends with the values -1, -1.

```
600 809 840 815 555 508 493 546 608 737 930
774 831 699 536 586 818 485 790 861 617 838
799 919 538 521 510 431 924 742 589 602 814
645 933 801 521 728 870 701 621 705 746 730
665 665 655 515 496 720 729 969 517 858 532
606 606 572 797 480 469 455 666 483 488 514
545 864 928 866 906 639 454 480 699 607 582
884 602 617 608 812 957 583 499 500 672 672
966 597 587 580 958 667 684 537 500 912 805
843 783 653 577 528 523 521 515 506 830 875
643 904 632 859 627 690 561 657 741 729 587
10 4
2 7
3 2
7 9
8 4
5 0
6 1
-1 -1
```

Program Output

The program must track the path of each particle in an 11x11 array. The array should be initialized with periods. Cells that are visited by a particle should be marked with an asterisk. After marking the cell, the program should examine the eight neighboring cells and move the particle to the cell with the smallest integer value in the corresponding input array. If there are no adjacent cells with lower values, or if the particle visits a cell that has already been visited, the program should begin tracking the next particle. The program should print the array after all the particles have been tracked.

```
.....*.....
......*....
...***.....
...........
........*.*
........**.
......**...
..*....*...
...*....*..
....****...
...........
```

Problem 12

Haunted House Inspector

10 points

JAVA: program name must be prob12.class
C /C++ program name must be: prob12.exe

Task Description

Every Halloween, the local fire department must inspect the charity-run Haunted House mazes, to ensure that the fire department's fire hoses are long enough to reach any potential fire within the attraction. To automate this process, you decide to write a computer simulation to show whether or not this is the case.

Program Input

Your program will read in a file (prob12.in) containing a scaled map (one character equals one foot), with simulated fires located throughout a maze. The first line of the file will contain three numerical values, the width of the maze (characters on one line), the length of the maze (number of lines of input), and the length of your fire hose. The maze has a maximum of 60 feet on a side. Sides of the maze and interior walls are marked with hash "#" signs. The entry to the maze will be marked with an "S" along an outside wall. Fire locations are represented by "F", and blank spaces fill the rest of the interior. Your hose must enter the building at the starting "S", and traverse through the maze until it reaches its maximum length. Corners are taken at right angles (no diagonals), and the hose must touch the fire location to put it out.

```
60 7 75
############################################################
#FFFFFF FF      # #FFF #         #       #   #FF    #FFFFFF#
############### # #### ####### # #### #### ######## #F######
S F F  FFFFF FFF#  FFFFFFF   # # #FFFF   #  F#      #FFFFFF#
##########F####F# ############ # ####### # ### ###########F#
#  FFF  FFFFF       FFFFFFFF   #                     FFFFFF#
############################################################
```

Program Output

Print out the building map showing the result of the simulation. Place a period (.) in each location that your hose could reach. If the fire was completely put out, also print out "All Fires Extinguished!".

```
############################################################
#...............#.#....#.........#.......#...#......#FFFFFF#
###############.#.####.#######.#.####.####.########.#F######
................#............#.#.#.......#...#......#F.....#
##########.####.#.############.#.#######.#.###.###########.#
#..............................#...........................#
############################################################
```

Problem 13

Least Recently Used

12 points

JAVA: program name must be prob13.class
C /C++ program name must be: prob13.exe

Task Description

We all want our computers faster, cheaper, and with more storage. The Least Recently Used algorithm provides computer designers with a way to increase system performance with little additional cost. It does this by allowing commonly accessed pages to be read from fast system memory instead of slow disk drives. Understanding how it works requires some explanation.

People commonly express memory and disk storage capacity in bytes or gigabytes, but memory is often organized into pages of 4KB each. Hard drives can permanently store far more pages of data than RAM, with a much lower cost per GB. Disk access, however, is significantly slower than RAM. If only we could have hard disk drives as fast as main memory! For now, at least, it is not so.

Let's consider a way to improve performance with the hardware we have. Studies have shown that computers often access some pages repeatedly from the disk, while most of the disk's pages are very rarely accessed. This access pattern has led computer designers to store commonly accessed pages in otherwise unused portions of memory called a cache. A cache requires an effective algorithm to achieve good results. When a program accesses a page from the disk drive, the system first looks to see if the page is in the cache. If it is, this is called a cache hit and the page is transferred from the cache to the program's memory. Very fast! If not, this is called a cache miss and the cache system must read the page from the disk. In addition, the cache system must decide what to do with this newly read page. A key feature of any caching algorithm is its page replacement policy. In our case, on every cache miss we will replace the least recently used page in the cache with the currently accessed page. Hit or miss, the currently accessed page becomes the most recently used page.

For this program we will consider a very simple storage and cache system. The disk capacity is 1,024 pages (numbered from zero) and there are 16 pages of memory available for the cache.

There's more than one way to implement an LRU algorithm. Write one.

Program Input

The input (from file prob13.in) will consist of exactly five lines. Each line will contain twenty page numbers separated by spaces. The program must interpret the numbers as page requests in order from left to right.

```
390 444 390 591 556 635 259 960 701 398 344 396 339 690 874 960 418 635 875 390
390 581 960 14 390 960 500 390 731 398 623 344 398 960 417 390 390 355 344 134
875 89 701 146 290 635 635 777 217 146 722 960 960 390 134 390 623 960 793 264
134 417 282 138 406 558 390 740 134 390 979 453 43 390 217 344 697 601 398 390
417 453 89 398 396 970 398 398 259 720 390 89 259 993 89 390 418 14 86 398
```

Program Output

After every 20 page accesses (i.e. after each line of input) the program must print the cumulative cache hit rate as a percentage of the total number of pages accessed. The program begins with an empty cache.

```
cache hit rate: 20.00%
cache hit rate: 40.00%
cache hit rate: 41.67%
cache hit rate: 38.75%
cache hit rate: 42.00%
```

Problem 14

Six Degrees of Kevin Bacon

13 points

JAVA: program name must be prob14.class
C /C++ program name must be: prob14.exe

Task Description

The trivia game Six Degrees of Kevin Bacon is based on the concept of the small world phenomenon. The premise of the game is that any actor can be linked, through their film roles, to actor Kevin Bacon in six links or less. The game requires a group of players to try to connect any film actor in history to the Kevin Bacon as quickly as possible and in as few links as possible.

Write a program to compute an actor's Bacon number from a limited movie database.

Program Input

The input file (prob14.in) is divided into two sections. The first section lists movies that will be used to calculate Bacon numbers. The second section lists the actors whose Bacon numbers the program must calculate. Section names appear in square brackets. Movie and actor names are written in CamelCase*.

```
[MOVIES]
BeautyShop QueenLatifah AliciaSilverstone AndieMacDowell AlfreWoodard KevinBacon DellaReese
Footloose KevinBacon LoriSinger JohnLithgow DianneWiest ChrisPenn SarahJessicaParker
JFK KevinCostner TommyLeeJones KevinBacon GaryOldman JohnCandy DonaldSutherland JoePesci LaurieMetcalf
APrarieHomeCompanion TommyLeeJones MarylouiseBurke WoodyHarrelson KevinKline
ThePinkPanther SteveMartin KevinKline JeanReno EmilyMortimer BeyonceKnowles
Dreamgirls JamieFoxx BeyonceKnowles EddieMurphy DannyGlover JenniferHudson JohnLithgow
Shooter MarkWahlberg MichaelPena DannyGlover KateMara EliasKoteas RhonaMitra
TheItalianJob MarkWahlberg CharlizeTheron DonaldSutherland JasonStatham MosDef
16Blocks BruceWillis MosDef DavidMorse JennaStern
LuckyNumberSlevin JoshHartnett BruceWillis LucyLiu MorganFreeman BenKingsley
BatmanBegins ChristianBale MichaelCaine LiamNeeson KatieHolmes GaryOldman
ThePhantomMenace LiamNeeson EwanMcGregor NataliePortman IanMcDiarmid
AttackOfTheClones EwanMcGregor NataliePortman HaydenChristensen ChristopherLee SamuelLJackson
SinCity JessicaAlba JoshHartnett RutgerHauer BruceWillis ElijahWood MickeyRourke
EternalSunshineOfTheSpotlessMind JimCarrey KateWinslet ElijahWood KirstenDunst GerryRobertByrne
BruceAlmighty JimCarrey MorganFreeman JenniferAniston PhilipBakerHall CatherineBell SteveCarell
TheTwoTowers SeanAstin CateBlanchett OrlandoBloom ChristopherLee ElijahWood IanMcKellen ViggoMortensen
TheAviator LeonardoDiCaprio CateBlanchett KateBeckinsale AlecBaldwin AlanAlda JudeLaw BrentSpiner
MenInBlack TommyLeeJones WillSmith LindaFiorentino VincentDOnofrio RipTorn TonyShaloub
WildWildWest WillSmith KevinKline KennethBranagh SalmaHayek MemmetWalsh
FunWithDickAndJane JimCarrey TeaLeoni AlecBaldwin AngieHarmon LaurieMetcalf JohnMichaelHiggins
Dogma BenAffleck MattDamon LindaFiorentino JasonMewes ChrisRock AlanRickman JasonLee SalmaHayek
[ACTORS]
EddieMurphy
[END]
```

Program Output

The program should display the Bacon number and the set of links used to compute the number.

```
EddieMurphy has a Bacon number of 2
EddieMurphy was in Dreamgirls with JohnLithgow
JohnLithgow was in Footloose with KevinBacon
```

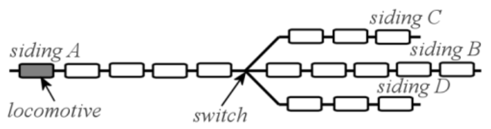
Problem 15

Railway Shunting Crew

15 points (+3 point bonus)

JAVA: program name must be prob15.class
C /C++ program name must be: prob15.exe

Task Description



A railway shunting crew must fulfill orders for the formation of departing trains from the rail cars in its sidings. There are four sidings, lettered A, B, C and D, with a limited number of positions that a rail car may occupy: five each for A and B, three each for C and D. The formation order will specify the sequence of rail cars to be connected to the locomotive that occupies the outmost position of siding A. The train will be formed from five of the eight cars sitting in the sidings in the order specified. Rail cars may only move to or from siding A. This means, for example, that a car cannot move directly from siding C to siding D. Instead, it must first move from siding C to siding A (where a free space must therefore be available), then to siding D. In other words, all valid moves must have siding A as the source or destination.
Write a program that prints a series of valid rail car movements that fulfills the formation order.

A three-point bonus will be awarded to the team(s) with the most efficient (i.e. fewest moves) average solution for all tested data sets.

Program Input

The input consists of three lines. The first two lines represent the initial configuration of rail cars on the sidings. The representation is logical, not physical. The logical siding layout looks like this:

```
AAAAA+BBBBB
  CCC^DDD
```

The + and ^ characters represent the switch. The third line represents the formation order.

```
L----+86321
  457^---
L12345
```

```
also try these:     L----+35841     L----+26381
                      267^---         457^---
                    L12846          L41732
```

Program Output

The program must print the siding configuration at each step from initial configuration to fulfillment, with a blank line between each step. Only one car may move per step. Many correct solutions are possible. It is also possible to return to a previously visited configuration through a long sequence of moves. However, revisiting a previous configuration is a waste of the shunting crew's time (never mind the judges) and will be considered an incorrect solution. In the final train configuration, one of the rail cars must occupy the + location of the switch. The switch must remain unoccupied in all other steps. The program should complete within 15-20 seconds. Programs that do not complete within 60 seconds on a judge's machine will be considered incorrect.

```
L----+86321
  457^---

L8---+-6321
  457^---

L----+-6321
  457^--8

L6---+--321
  457^--8

L----+--321
  457^-68

L7---+--321
  45-^-68

L----+--321
  45-^768

L5---+--321
  4--^768

L54--+--321
  ---^768

L543-+---21
  ---^768

L54--+---21
  3--^768

L542-+----1
  3--^768

L54--+----1
  32-^768

L541-+-----
  32-^768

L54--+-----
  321^768

L547-+-----
  321^-68

L54--+----7
  321^-68

L5---+---47
  321^-68

L----+---47
  321^568

L1---+---47
  32-^568

L12--+---47
  3--^568

L123-+---47
  ---^568

L1234+----7
  ---^568

L12345----7
  ---^-68

23 moves.
```

Problem 16

Warning: Graphic Functions!

17 points

JAVA: program name must be prob16.class
C /C++ program name must be: prob16.exe

Task Description

Often times seeing the graph of a function is the best way to understand its properties.
Write a program to graph a function *y = f(x)* over a specified range.

Program Input

The program will read one or more lines typed from the console. Each line contains an algebraic expression of a single variable x. Valid operations include addition, subtraction, multiplication, division, exponentiation (where a^b is represented by `a^b`), parentheses, sine, cosine, and log10. The program must evaluate the expression using standard arithmetic precedence rules: first functions and parenthesis, then exponents, then multiplication and division, then addition and subtraction. Back-to-back exponents must be evaluated right-to-left, all other operations left-to-right. When correctly parsed, the input will not cause divide by zero conditions. Trigonometric functions should operate in radians, not degrees. The program must display a prompt to the console before each line with the string "f(x)=", and it must terminate when the input-line is the word END.

```
f(x)=7.2*cos( (x + 3) / 4 )
f(x)=log( x+21 ) * 5.771
f(x)=3 * x^2 -4 *x+ 1
f(x)=END
```

Program Output

The program must plot each function over the range -10 to 10 in a 21x21 integer grid. Output values should be truncated (always round down). Each function must be plotted in its own grid. The program must draw X and Y axes.

```
..........|..........
..........|..........
..........|..........
.......*..|..........
.....**.**|..........
....*.....*..........
..........|..........
...*......|*.........
..*.......|.*........
..........|..........
-*--------+--*-------
*.........|...*......
..........|....*.....
..........|..........
..........|.....*....
..........|......*...
..........|.......*..
..........|........**
..........|..........
..........|..........
..........|..........
```