codewars.5

Program / Task Name:	**Introduction**
Level:	**Novice**
Point Value:	**1**

Task Description

Computer programmers must learn some rudimentary social skills, even if their interpersonal interaction is constrained by digital media. Please write a program to introduce your team to the judges.

Program Input

Your program should print a prompt for the judge to enter his or her name like the bold face type below. Then it should read the name from the keyboard input. You may assume that the judge will enter one name only, so you don't have to worry about spaces in the input.

```
Please enter your name: Moldwart
```

Program Output

After reading the judge's name, your program must complete the introduction by (1) greeting the judge, and (2) displaying the name of your school, your mascot, and the names of each team member. You may use the format below as an example:

```
Hello, Moldwart!
We are Athens High School Lions: Mike Davis, Anh
Nguyen, and Isok Patel.
```

codewars.5

Program / Task Name:	**Height of a Rainbow**
Level:	**Novice**
Point Value:	**2**

Task Description

How tall is a rainbow?

Because of the way in which light is refracted by water droplets, the angle between the level of your eye and the top of the rainbow is always the same. If you know the distance to the rainbow, you can multiply it by the tangent of that angle to find the height of the rainbow.

The magic angle is 42.3333333 degrees.

The C++ standard library works in radians, so you may have to convert the angle to radians with the formula:

Radians = degrees x pi/180 (where pi = 3.14159265)

Through the header file cmath, the C++ standard library provides a tangent function named tan. This is a value returning function that takes a floating-point argument and returns a floating-point result.

```
x=tan(someAngle);
```

To determine the height of the rainbow, multiply the tangent by the distance to the rainbow.

Write a program to determine the height of a rainbow.

Program Input

Please prompt the judge to enter the distance to the rainbow (in meters) as shown below:

```
Enter the distance to the rainbow (in meters): 52
```

You can assume that:

- The value entered will be a number, and
- The value will be an integer (the judge will not enter a decimal number, i.e. 52.3)

Program Output

After reading the distance to the rainbow, determine the height of the rainbow and print it out to 4 decimal places.

```
The height of the rainbow is 47.3717 meters.
```

codewars.5

Program / Task Name:	**Word Scramble**
Level:	**Novice**
Point Value:	**3**

Task Description

Write a program that will reverse the letters in each of a sequence of words while preserving the order of the words themselves.

Program Input

The text to scramble can be found in prob03.in and might look as follows:

```
2002 noititepmoC gnimmargorP sraW edoC
2002 ,2 hcraM
noitaroproC retupmoC qapmoC
melborP elbmarcS droW
```

Program Output

Write the scrambled text to the screen.

```
Code Wars Programming Competition 2002
March 2, 2002
Hewlett-Packard Company
Word Scramble Problem
```

codewars.5

Program / Task Name:	**Hangman**
Level:	**Novice**
Point Value:	**4**

Task Description

Create a computerized version of the game of “hangman”. The judge will specify both the solution for the game and the letter guesses for the game. Your program must determine if the judge has won or lost the game (based on the letter guesses) and draw the traditional hangman picture as follows:

```
1st Wrong Guess: “ O” <<Carriage Return>>
2nd Wrong Guess: “/ “
3rd Wrong Guess: “| “
4th Wrong Guess: “ \” <<Carriage Return>>
5th Wrong Guess: “ / “
6th Wrong Guess: “ \” <<Carriage Return>>
```

Some languages use the “\” character as a control character. If you’re using one of the languages please make the appropriate adjustments in the code (i.e. C and C++ use this “\\” to represent this “\”).

Program Input

Your program should print a prompt for the judge to enter a solution. Then prompt the judge for the letter guesses (all on 1 line). You can assume that:

- All solutions will be either 1 or 2 words,
- No solution will be longer than 20 letters in length (including the space between the words),
- The judge will specify enough letter guesses to either win or lose the game,
- All letter guesses are case insensitive (treat ‘a’ and ‘A” as the same letter), and
- No numbers will be entered,
- No letter guess will be entered more than once,
- Only 1 solution will be specified per game (running of the program).

Since the judge is specifying all of the letter guesses on a single line, ignore all letter guesses after the 6th incorrect guess.

```
Please enter solution: Wayne Gretzky
Please enter your guesses: aeioubtwgsynzkr
```

Program Output

After reading both the solution and letter guesses, your program must determine if the judge has won (You Won!) or lost (You Lost!) the game and print out the current state of the hangman.

```
You Won!
 O
/|\
```

/

codewars.5

Program / Task Name:	Unique Word Count
Level:	Novice
Point Value:	4

Task Description

Ever wonder how many unique words are in a section of text?

Write a program to count the number of unique words in a section of text.

There are two special rules to consider while creating your program:

- Words separated by a hyphen should be counted as one word (self-employed should be counted as one word), and
- The case of a letter does not matter (The and the are considered the same word, and would only count once towards the total word count).

You can also assume that:

- The section of text in prob05.in will not be longer than 10,000 characters, and
- No single word will be longer than 80 characters.

The only punctuation used are periods (.), commas (,), question marks (?) and semicolons (;)

Program Input

The section of text to process is in prob05.in and might look something like this:

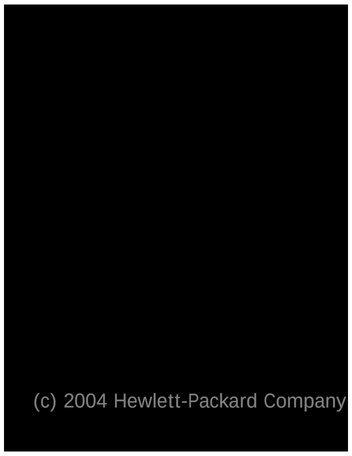
```
Fourscore and seven years ago our fathers brought
forth on this continent a new nation, conceived in
liberty and dedicated to the proposition that all
men are created equal. Now we are engaged in a
great civil war, testing whether that nation or
any nation so conceived and so dedicated can long
endure. We are met on a great battlefield of that
war. We have come to dedicate a portion of that
field as a final resting-place for those who here
gave their lives that that nation might live. It
is altogether fitting and proper that we should do
this. But in a larger sense, we cannot dedicate,
we cannot consecrate, we cannot hallow this
ground. The brave men, living and dead who
struggled here have consecrated it far above our
poor power to add or detract. The world will
little note nor long remember what we say here,
but it can never forget what they did here.  It is
for us the living rather to be dedicated here to
the unfinished work which they who fought here
have thus far so nobly advanced. It is rather for
us to be here dedicated to the great task
remaining before us; that from these honored dead
we take increased devotion to that cause for which
they gave the last full measure of devotion; that
we here highly resolve that these dead shall not
have died in vain, that this nation under God
shall have a new birth of freedom, and that
```

```
government of the people, by the people, for the
people shall not perish from the earth.
```

Program Output

After processing the section of text, output the number of unique words to the screen as follows:

```
There are 137 unique words.
```



codewars.5

Program / Task Name:	Quench Your Thirst
Level:	Intermediate
Point Value:	5

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Task Description

The local convenience store is having problems keeping its soda machine stocked. They've asked you to write a program to simulate the process of buying soda from the machine. The cost of a soda is 55 cents and you will accept nickels, dimes and quarters. After a customer has inserted enough money and selected a soda, give them a soda and return their change (if any) in the least amount of coins possible.

You can assume:

- The customer will complete all transactions they start (in other words, once the first coin goes in, they're committed),
- Only one soda will be purchased per transaction,
- The machine is fully stocked with (Coke, Sprite, Dr. Pepper) and it will not run out, and

The machine is fully stocked with change (nickels, dimes, and quarters) and it will not run out.

Program Input

Prob06.in contains the initial contents of the machine (soda and change), as well as all of the data for the soda purchases for a single day. A sample input file, might look as follows:

```
64 48 32
100 400 300
C 25 25 25
S 10 10 25 5 5
D 25 25 25 25
C 10 5 10 5 10 5 10 5
C 25 25 10
C 25 25 5 5 5
S 25 25 25 25 25 25
C 10 10 10 10 10 10 10
D 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
S 10 25 5 10 25 5
C 25 25 25
D 25 25 25
C 25 25 10 10
C 25 25 10
C 10 25 10 10
S 25 25 5 5 5 5
```

The first line is the number of Cokes (64), Sprites (48), and Dr. Peppers (32) in the machine.

The second line is the number of quarters (100), dimes (400), and nickels (300).

All subsequent lines contain the information for purchasing a soda:

- The first parameter is the type of soda (C = Coke, S = Sprite, D = Dr. Pepper), and

- The rest of the parameters are the coins inserted into the machine (25 = quarter, 10 = dime, 5 = nickel).

Program Output

Store all results in prob06.out according to the following specifications:

- Before starting the simulation create an inventory report of the machine:
  - Total number of sodas in the machine,
  - Number of each type of soda in the machine (Coke, Sprite, Dr. Pepper),
  - Total amount of money in the machine, and
  - Number of each type of coin in the machine (quarters, nickels, dimes).
- Print out a line stating that the simulation of purchasing soda has begun,
- Report each purchase of soda using this format:
  - Type of soda purchased (Coke, Sprite, Dr. Pepper)
  - Amount of money inserted
  - Amount of change returned
  - A list of each coin returned
- After all of the purchases have been completed, print out another line stating that the simulation of purchasing soda has completed,
- After all of the purchases have been completed, create an inventory report of the machine exactly like the report before you started your simulation

For the data specified above, the report would look as follows:

```
Number of sodas: 144 (64 Cokes, 48 Sprites, 32 Dr.
Peppers)
Amount of money: $80.00 (100 Quarters, 400 Dimes,
300 Nickels)

Begin purchasing inserting money and purchasing a
soda.

Deposited   Soda        Change
---------   ----        --------------------------
-
$0.75       Coke        $0.20 (2 dime)
$0.55       Sprite      $0.00
$1.00       Dr. Pepper  $0.45 (1 quarter, 2 dime)
$0.60       Coke        $0.05 (1 nickel)
$0.60       Coke        $0.05 (1 nickel)
$0.65       Coke        $0.10 (1 dime)
$1.50       Sprite      $0.95 (3 quarter, 2 dime)
$0.70       Coke        $0.15 (1 dime, 1 nickel)
$1.00       Dr. Pepper  $0.45 (1 quarter, 2 dime)
$0.80       Sprite      $0.25 (1 quarter)
$0.75       Coke        $0.20 (2 dime)
$0.75       Dr. Pepper  $0.20 (2 dime)
$0.70       Coke        $0.15 (1 dime, 1 nickel)
$0.60       Coke        $0.05 (1 nickel)
$0.55       Coke        $0.00
$0.70       Sprite      $0.15 (1 dime, 1 nickel)

All purchases have been processed

Number of sodas: 128 (55 Cokes, 44 Sprites, 29 Dr.
Peppers
```

```
Amount of money: $88.80 (127 Quarters, 406 Dimes,
329 Nickels)
```

codewars.5

Program / Task Name:	Shooting Star
Level:	Intermediate
Point Value:	6

Task Description

The Really Simple Game Company has just hired you. Your first programming assignment is to write the “Shooting Stars” game. This game is played on a 3 X 3 grid. Each location on the grid has either a star or no star. An asterisk represents a “star” and a period represents “no star”. Each location on the grid is numbered 1-9, going from left to right then top to bottom.

Grid Mapping

1	2	3
4	5	6
7	8	9

The object of the game is to destroy all of the stars. To do this you shoot star bombs at the grid, until there are no stars left. You shoot a star bomb by typing the number of the grid location. For any location you shoot at and the ones adjacent vertically and horizontally, if there is a star there it disappears and if there is no star there it creates one. There is one star in the center for the start of the game and looks like this:

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

If the first character is a ‘5’, the resulting grid would look like:

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

Program Input

Any key character can be input, but for a shot action all are ignored except numeric digits 1-9.

Program Output

At the beginning of the program print out “Welcome to Shooting Stars”, a blank line, the initial star grid and then another blank line. Do not print out any characters entered. Any character entered that is not a 1-9 causes should be ignored. After a valid numeric digit is input, print out the new playing grid of stars with a blank line after the star grid. If all stars have been destroyed, then print out “You WIN!!!” and exit the program.

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

codewars.5

Program / Task Name:	What's Your Average?
Level:	Intermediate
Point Value:	7

Task Description

Suzy Schoolteacher has always tried to be as fair as possible when judging her students at the end of the year. She will drop the lowest (only one) homework grade from each student's list of homework grades. In the case of Alex Smith, she would drop the 68 because it is his lowest homework grade.

Create a grade-book program that reads in all the students' names and grades, and then computes their final averages using Suzy's grading techniques. Homework grades are preceded by an 'H' and test grades are preceded by a 'T'.

- The number of students and the number of grades is undetermined (but will not exceed 50).
- All students will have the same number of grades.
- A student's final grade is 60% of his/her test average and 40% of his/her homework average.
- Round all final grades to the nearest tenth (83.65 would round to 83.7).

Program Input

The grades for the grade book can be found in prob08.in and will look as follows:

```
11 5
Alex Smith H 75 88 94 95 84 68 91 74 100 82 93 T
73 82 81 92 85
Susan Wright H 86 55 96 78 93 85 80 74 76 82 62 T
82 89 93 70 74
John Jones H 84 66 74 98 92 85 100 95 96 42 88 T
88 94 100 82 95
Jane Doe H 73 99 98 83 85 92 100 60 74 98 92 T 84
96 79 91 95
Jimmy Johnson H 65 72 78 80 82 74 76 0 85 75 76 T
74 79 70 83 78
```

Program Output

Write the averages for each student into prob08.out as follows:

```
Averages
Alex Smith 84.6
Susan Wright 81.4
John Jones 90.2
Jane Doe 89.2
Jimmy Johnson 76.6
```

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

[redacted]

codewars.5

Program / Task Name:	**Movie Time Popcorn**
Level:	**Intermediate**
Point Value:	**7**

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Task Description

The "Movie Time Popcorn" farm cooperative is about to sign a deal to supply all of the popcorn for a nationwide movie theatre chain. If the deal goes through, it would make the farm cooperative a major success. The last holdup to the deal is that the movie theatre wants proof that the farm cooperative can deliver the popcorn it needs. The farm cooperative has hired you to create a program to produce a bar chart of the popcorn production for a cooperative farm group on a farm-by-farm basis.

The cooperative has supplied you with a data file containing the information on each farm (name, number of acres, and number of pints of popcorn produced).

You can assume that:

- No farm name will be longer than 25 characters,
- Each line will contain only 1 set of data, and
- Each farm will have a full set of data (name, number, pints of popcorn)

Each item of data for a farm will be separated a comma and a space.

Program Input

The data for the farms can be found in prob09.in, and will look as follows:

```
     Orville's Acres, 114.8, 438010
     Hoffman's Hills, 77.2, 362290
     Jiffy Quick Farm, 89.4, 248120
     Jolly Good Plantation, 183.2, 1045700
     Organically Grown Inc., 45.5, 146830
```

Program Output

Produce a single bar chart for all of the farms in the cooperative and write it to prob09.out. For each farm, create a single line on the bar chart containing the name of the farm and it's production of popcorn per acre. Mark the bar chart with a '*' for every 250 pint jars of popcorn per acre. The production goal for each farm is 5000 pint jars per acre. Since the name of the farm can't be longer than 25 characters, start the bar chart in column 30.

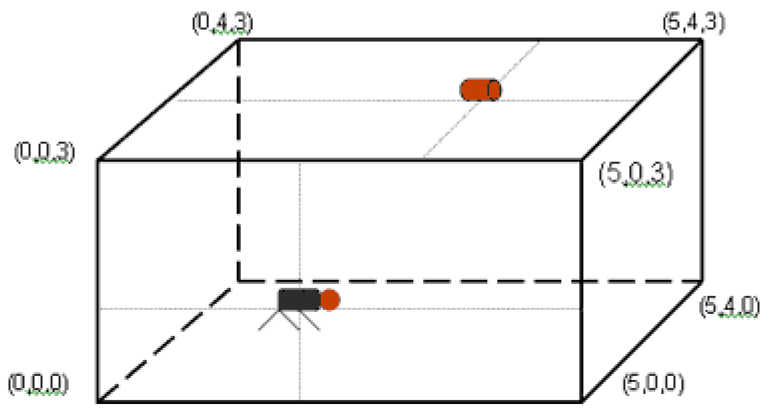
```
                   Pop Co-Op

Farm Name                    Production in
                             Thousands of
                             Pint Jars per Acre
                                1   2   3   4   5   6
                             ---|---|---|---|---|---|
Orville's Acres              ***************    |
Hoffman's Hills              ****************** |
Jiffy Quick Farm             ***********        |
Jolly Good Plantation        *******************#**
Organically Grown Inc        ************       |
```

codewars.5

Program / Task Name:	The Ant and the Candy
Level:	Intermediate
Point Value:	9

Task Description

There is an ant and a candy on the surface of a rectangular box. The ant cannot fly. It can only walk on the surface of the box to the candy. Please write a program that calculates the distance of the shortest path from the ant to the candy. The coordinates of the eight corners of the rectangular box are shown in the figure below.



The initial position of the ant can be on the top or one of the sides of the box (including the corners). It cannot be on the bottom. The candy is always on the top of the box. The candy cannot be on the edges or corners of the box (or it will fall off).

Program Input

The input to the program is provided in prob10.in. Each input line consists of six numbers separated by a blank. First three numbers are the coordinates representing the position of the ant while the remaining three numbers are the coordinates representing the position of the candy. You may assume these coordinates are on the surface of the rectangular box. For example, the first line of the sample input means that the ant is at (3,1,3) and the candy is at (3,3,3). Assume that the data given is always valid.

Sample Input:

```
3 1 3 3 3 3
2.25 0 2 2.5 2 3
0 0 0 5 4 3.0
0 4 3 5 0.0 3
5 0 3 5 4.00 3
```

Program Output

A real number represents the distance of the shortest path from the ant to the candy. On the screen, print the distance accurate to two digits to the right of the decimal point in the following format.

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Sample Output:

```
Shortest distance = 2.00 units
Shortest distance = 3.01 units
Shortest distance = 8.60 units
Shortest distance = 6.40 units
Shortest distance = 4.00 units
```

codewars.5

Program / Task Name:	**English Math**
Level:	**Advanced**
Point Value:	**11**

Task Description

Create a program that can process simple math problems that are stated in full English words and provide the answer (also in English). Keep the following in mind when writing your program:

- Only the numbers one through twenty will be used for inputs or outputs,
- Only the operators "times", "divided-by", "plus", and "minus" will be used,
- Input and output should be all in lowercase letters,
- Correct answers will be integer numbers from one to twenty - only integer math operations should be used (so fractions or rounding should not be an issue) and intermediate results can go outside the range of 1-20,
- The input file (prog11.in) may have up to twenty lines of input,
- Each line will contain a different math problem,
- There may be more than one operation per line/problem (maximum of 10 operations per line),
- The operators have the following order of precedence (when there is more than one operator per line/problem):
  1. "times"
  2. "divided-by"
  3. "plus"
  4. "minus"

  A higher precedence means that all operations of that level are performed before ones of the next level.

Answers must be output on separate, consecutive lines (see example below) and correct spelling is required.

Program Input

The input for your program can be found in prob11.in and looks as follows:

```
two plus three
five times four minus three
twelve minus six
four divided-by two times two
three times four minus four divided-by two
seven minus one plus twenty divided-by two times
five
```

Program Output

Write the output to a file (prob11.out) as follows:

```
five
seventeen
six
one
```

```
ten
four
```

codewars.5

Program / Task Name:	Server Blade Power Consumption
Level:	Advanced
Point Value:	13

Task Description

Server blades are the newest trend in Web Server design. A web site may be built on a large number of very compact ‘blade’ servers, which omit the expansion slots, floppy and CD-ROM drives in order to squeeze the server into a very small package. The blades fit into an “Enclosure”, which allows up to 20 blade servers to share common parts (cooling fans, power supplies, etc.). Each blade server consumes less power than a traditional web server, but since upwards of 300 blade servers fit into a single computer rack, power consumption is still an important issue!

In addition, blade servers include power-saving modes, which allow them to reduce their power usage during off-peak hours. This is the same technology used in laptop computers to squeeze more time out of their batteries. A blade normally operates in FULL power mode, providing the most performance available. When the server is not busy, it will go to LOW power mode, reducing performance but saving electricity. If the server is not needed for a while, it can go into SLEEP mode, which turns the server off, but allows it to ‘wake up’ quickly when needed (servers consume a little bit of power while in sleep mode).

Your task is to build a reporting system which, given a one-day log file showing changes in power usage, will generate a daily power and system uptime report for an enclosure of 20 blade servers.

Each blade server can be identified by its location within the enclosure (each blade server occupies a single “bay”, or slot, in the enclosure). These bays are numbered from 1 to 20. A blade’s power consumption can be calculated by multiplying its power consumption rate by the time spent operating at that rate. Power consumption is measured in Watt-hours, which we commonly abbreviate as just “Watts”. For example, your desk lamp probably has a 60 Watt light bulb (it’s really a 60 Watt-hour light bulb). If you run the lamp for 2 hours, you’ve consumed 120 Watts. You buy power from the electric company in units of kilowatt-hours. If power costs $0.09 per kilowatt-hour, running that lamp for 2 hours cost about a penny (.120 kilowatt-hours at $.09 per kilowatt-hour equals $.0108).

Since computers convert all electrical energy they consume into waste heat, we can also calculate how much heat is produced by the blades. This information is important to computer room designers, as it tells them how big their air conditioning equipment needs to be. Heat output is measured in BTU’s per hour (again, we usually just say BTU’s). 1 Watt-hour of energy converted to heat will generate 3.412 BTU/hr. So that desk lamp would have generated 409.44 BTU’s while it was on.

Program Input

The input file consists of two header lines, followed by the Power Event Log. Commas separate each field in the input file. The first header line contains a report date (MM/DD/YYYY), the customer’s company name, and the cost of

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

electricity per kilowatt-hour (in dollars and cents). The second header line describes the power consumption ratings of a single blade in its four operating modes: OFF, SLEEP, LOW, and FULL.

The Power Event Log contains a list of each power event, listed on its own line. Each event contains three fields, separated by commas. The first field is the time of the event, in the format HH:MM, in 24-hour time (for example, 12:39am is stored as 00:39, while 2:00pm is stored as 14:00). This is followed by the blade's bay number, and lastly by the new power state of the blade. For example, the line "00:01,1,FULL" means that the blade in bay #1 went to FULL power mode at 12:01am.

You should continue reading Power Event Log entries until you reach the end of the file. The Power Event Log could contain hundreds or even thousands of entries.

The log should be interpreted using the following rules:

- The log always starts at 00:00 (midnight), although no events are guaranteed to occur at midnight
- Before the start of the log, all server blades are OFF
- Log entries will always occur ascending in time (time never goes backwards!)
- Log entries occurring within the same minute can be assumed to occur simultaneously (at the start of each minute)
- The log ends at 00:00 on the next day, and the last event could occur at 23:59 (11:59pm)

A simple example input file is shown below (for a system with 5 blades).

```
01/31/2002,WebVan.com,0.08
0.00,2.35,14.90,24.60
00:01,1,FULL
00:01,2,LOW
00:01,3,SLEEP
00:07,4,SLEEP
00:39,5,SLEEP
01:45,1,LOW
04:20,2,SLEEP
|08:55,1,FULL
09:12,2,FULL
09:12,3,FULL
09:12,4,FULL
18:30,5,OFF
```

Program Output

The program must respond by writing a report to the output file. The report header must include a report title, the date of the report, and the customer's name. The main report section must list each blade (1-20) on a separate line, with each line containing the bay number, the total uptime for the blade in that bay (HH:MM), and the total power consumed by the blade (in Watts, to two decimal places). Uptime counts the number of minutes a blade server is in either the FULL or LOW power states.

At the end of the report, four lines must appear that show the following information for all 20 blades: The total uptime (HH:MM), the total power consumed, heat produced, and cost of the electricity used (all to two decimal places).

Do not round any values until ALL of the calculations have been made. The following values must fall within the specified error ranges:

Power Consumed (by a single blade): +/- .5 Watt
Total Power Consumption: +/- 5 Watts
Total Heat Produced: +/- 5 BTUs
Total Electrical Service Cost: +/- $0.02

See the results below for examples.

```
              Server Blade Power Log
              January 31, 2002
              Enclosure: Webvan.com

       Bay    Uptime        Power Consumed
       ---    ------        --------------
       1      23:59         520.46 Watts
       2      19:07         439.84 Watts
       3      14:48         385.68 Watts
       4      14:48         385.44 Watts
       5      00:00         41.98 Watts

       Total combined uptime: 72:42
       Total power consumption: 1773.40 Watts
       Total heat produced was 6050.84 BTUs

       Total electrical service cost was $0.14
```

codewars.5

Program / Task Name:	Stereo Hookup
Level:	Advanced
Point Value:	13

Task Description

Hooking up a home theater system can be difficult and confusing. To achieve the best audio and video quality, each source device (such as a DVD player, VCR, or tape deck) must be connected to the receiver (the central ‘hub’, which controls volume and allows switching from one device to another) using the best interconnect method available. For audio signals, a Coax digital connection is preferred, followed closely by an Optical digital connection, and lastly an analog connection. For video signals, a Component video connection is preferred, followed closely by an S-Video connection, and lastly a Composite connection.

Receivers have several inputs to support the connection of a large number of devices (5 or more inputs is typical). Each input may support more than one type of connection (for both audio and video), but to keep down the price of the receiver, most manufacturers limit the quantity of each connection type. For example, a typical receiver contains only two inputs with Component video connectors, while it has five or more inputs with Composite video connectors. This makes sense because not every device supports all of the connection types. For example, a VCR will only provide Composite video connectors, but a DVD player will provide Component, S-Video, and Composite video connectors.

Your task is to create a program to optimize the cable connections to a home theater receiver, by matching each device to the best available input on the receiver, and list instructions showing the device connections and all the cables required.

The source devices must be connected using the following rules:

- Devices must be connected using the best available connection to a receiver input. Video connections take priority over audio connections (that is, a receiver input selection should be made using the video portion first, and the audio portion second).

- If the best matching audio or video connection type is not available, the device must be connected using the next best available type. For example, if no Component video connections are available, a typical DVD player would be connected using an S-Video connection.

- Any device supporting video must have both an audio and a video connection. If no video inputs are available, the device cannot be connected to the receiver.

- Only one audio and one video cable are required for each device. For example, a device connected using a Coax digital connection does not need an Optical digital or Analog cable.

- Devices with the most capabilities take priority over devices with fewer capabilities. For example, an audio device with a Coax digital output must be connected (even if it means using the lesser-quality Analog audio connection) before an audio device with only an analog connection. Similarly, devices with video capability take priority over audio-only devices.

- If two devices have identical capabilities, the device appearing first in the input file takes priority.

Only one device can be connected to each receiver input. If the number of devices exceeds the number of inputs, then the lowest priority device(s) cannot be connected.

Program Input

The input file consists of a receiver section, and a source devices section. The first line of the receiver section contains two strings: the keyword RECEIVER, and the product name of the receiver (maximum of 20 characters). The next lines of the receiver section list each receiver input and its capabilities, one receiver input per line. Each line starts with the name of the receiver input (maximum of 8 characters), followed by the audio connection types available, followed by the available video connection types. The audio connection types will always be in the following order: ANALOG, OPTICAL-DIGITAL, COAX-DIGITAL. The video connection types will always be in the following order: COMPOSITE, S-VIDEO, COMPONENT-VIDEO. Field names are separated by commas. The maximum number of receiver inputs is ten. The receiver section ends with the keyword END-RECEIVER on a separate line.

The device section contains a list of devices to be attached, one device per line. Each line starts with the keyword DEVICE, followed by the name of the device (maximum of 20 characters), followed by the audio connection types, followed by the video connection types (in the same order defined in the receiver section). The device section ends with the keyword END-DEVICE on a separate line.

A very simple example input file is shown below.

```
RECEIVER, Acme RJX-3900
CD, ANALOG, COAX-DIGITAL
VCR1, ANALOG, COMPOSITE
VCR2, ANALOG, COMPOSITE
DVD, ANALOG, COAX-DIGITAL, COMPOSITE, S-VIDEO
END-RECEIVER

DEVICE, El Cheapo CD, ANALOG
DEVICE, Pannasonik VCR, ANALOG, COMPOSITE
DEVICE, Suny DVD, ANALOG, COAX-DIGITAL, COMPOSITE,
S-VIDEO
DEVICE, Disco Starr 8-Track, ANALOG
DEVICE, Hello Kitty Cassette, ANALOG
END-DEVICE
```

Program Output

The program must respond to each query by writing a report to the output file. The report title must include the receiver's product name. It must show the steps required for connecting the devices to the best receiver input. Each numbered step must describing a connection needed (from device to receiver input), and must be reported in the same order as the devices appeared in the input file. Finally, the report must conclude with a list of the cables required to complete the installation, including both the quantity and type of cable. See the results below for examples.

```
Home Theater Receiver Connections for: Acme RJX-
3900

1. Connect the El Cheapo CD audio to the CD input
using an Analog audio cable
2. Connect the Pannasonik VCR audio to the VCR1
input using an Analog audio cable
3. Connect the Pannasonik VCR video to the VCR1
input using a Composite video cable
4. Connect the Suny DVD audio to the DVD input
using a Coax-Digital audio cable
5. Connect the Suny DVD video to the DVD input
using an S-Video video cable
6. Connect the Disco Starr 8-Track audio to the
VCR2 input using an Analog audio cable

NOTE: The Hello Kitty Cassette cannot be connected
to this receiver

The cables required to complete this installation
are as follows:

Qty   Cable Type
---   --------------------
1      S-Video video
1      Composite video
1      Coax-digital audio
3      Analog audio
```

codewars.5

Program / Task Name:	**Wireless Network**
Level:	**Advanced**
Point Value:	**14**

Task Description

The iPAQ Pocket PC from Compaq can access local area networks (LANs) using an 11Mbps wireless LAN card. The card communicates with wireless access points located in and around the LAN. Your task is to write a program to test the coverage area of a set of wireless access points.

To solve this problem you will need to consider that the wireless access points have a limited range of communication which may vary from one model to the next. This means that most WLAN installations will contain more than one wireless access point. Therefore your program will need to read the WLAN installation as a set of (x,y) locations of wireless access points, each with its own service radius.

You will also need to consider the purpose of using a wireless LAN: to provide mobile access to the network. Therefore the test conditions for the WLAN will be straight line-segment paths. Each path from one (x,y) position to another (x,y) position represents a person using an iPaq moving from one point to another.

For each path, your program must determine if the person encounters a "dead" spot - a spot not serviced by any of the wireless access points.

Program Input

The input file contains two sections. Section one contains a list of all the access point locations. Each access point has an (x,y) location and a service radius. Section two is a list of (x,y) pairs indicating path start and stop.

```
ACCESS
198,202 150
300,500 300
700,500 150
800,200 150
PATHS
160,420 160,380
460,580 680,580
782,139 758,421
404,306 850,253
```

Program Output

The output file should list each path (x,y) pair followed by either the phase "DEAD SPOT" or "ACCESS CLEAR".

```
160,420 160,380 ACCESS CLEAR
460,580 680,580 ACCESS CLEAR
782,139 758,421 ACCESS CLEAR
404,306 850,253 DEAD SPOT
```

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

codewars.5

Program / Task Name:	**The Cosmic Milkman**
Level:	**Advanced**
Point Value:	**16**

hp codewars

back to code wars V

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Task Description

Long, long ago, in a galaxy far, far away…

A milkman sets forth in his starship to makes his rounds to all the planets in his solar system.

His solar system was unique such that all the habitable planets orbited in the same plane, giving the solar system a 2-Dimensional look. His solar system was also very unique in that all the habitable planets moved through space at the same uniform speed of 50,000Km per hour, either clockwise or counter-clockwise around the sun (or vice versa if you're looking at the solar system from the opposite direction), with the only difference being their distance from the sun and their locations in the arc of their orbits.

At an arbitrary time 0, all the planets are located as follows: distance from the sun (the center of the system); degrees into the arc of orbit around sun relative to an arbitrary positive x-axis (see picture below) counting in a counter-clockwise direction; and the direction of orbit:

Planet X – 1,000,000Km; 90 degrees; moving clockwise

Planet Y – 5,000,000Km; 50 degrees; moving clockwise

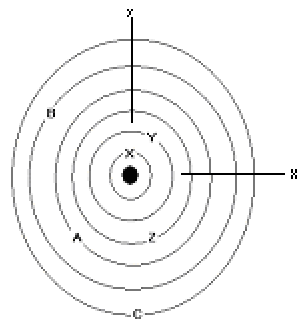
Planet Z – 10,000,000Km; 290 degrees; moving counter-clockwise

Planet A – 20,000,000Km; 230 degrees; moving counter clockwise

Planet B – 50,000,000Km; 140 degrees; moving clockwise

Planet C – 120,000,000Km; 275 degrees; moving counter-clockwise

Here is a rough sketch of the solar system at time zero (distances not to scale):



Other things to consider: our traveling salesman's spaceship can only fly at a maximum speed of 60,000Km per hour and assume that it takes him practically no time to drop off his shipment of milk once he arrives at a planet. Consider the constant pi = 3.141592654. Assume locations given for each planet are for a point at the center of each planet and ignore the diameter of the sun and the planets when calculating distances. Math Hint: the x and y coordinates of a planet are can be determined using rcosΘ and rsinΘ respectively.

Your job is to write a program that calculates the minimum total length of time it takes the milkman to travel a given route. Be careful to be exact. A correct answer must be within one hour of the judges' calculated time.

Program Input

The input for the problem will be provided as arguments on the command line when the program is executed as follows:

```
prob15 2500 X A B Z C Y X
```

- Arg1: is the time (in hours) relative to time 0 mentioned above that the salesman begins his trip from his home planet. Do not count this time in the total trip time.
- Args 2 – 8: are the list of planets, in the order that the milkman visits them, ending at his home planet. There will be exactly seven "hops" to the milkman's trip in order to visit all the other planets and return home. The home planet may vary from the example below.

Program Output

Print out the minimum time in hours that the trip will take as shown below:

```
The minimum numbers of hours for the trip is: 6121
```

Although the judging input will vary from this example's input, this example can be used for verifying your program prior to submitting to the judges. Here are some more details for this example:

Leg Time(mins): 20909 From X To A
Leg Time(mins): 45743 From A To B
Leg Time(mins): 47260 From B To Z
Leg Time(mins): 124597 From Z To C
Leg Time(mins): 124752 From C To Y
Leg Time(mins): 4016 From Y To X
Total Time(mins): 367277
Total Time(hrs): 6121