codewars.6

Program / Task Name:	Code Warrior 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: Joe
```

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

back to code wars VI

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

codewars.6

Program / Task Name:	**Compound Interest**
Level:	**Novice**
Point Value:	**2**

Task Description

You have just graduated from college with a degree in Computer Science and you've landed a GREAT job as a developer at Hewlett-Packard. Only problem is, you don't have a car to get back and forth from work.

Write a program to determine your monthly car payment. The formula for determining the monthly payment of a loan is:

$$P = L[c(1 + c)n]/[(1 + c)n - 1]$$

P = monthly payment
L = loan amount in dollars
n = number of months for the loan
c = the monthly interest rate (for example, if the yearly interest is 6%, c is 0.06/12 or 0.005)

Program Input

Your program should prompt for the cost of the car (an integer value), down payment (an integer value), yearly interest rate (a decimal value), and number of months for the loan (an integer value).

```
Enter the cost of the car: 25000
Enter the down payment: 5000
Enter the yearly interest rate: 5.9
Enter the number of month for the loan: 48
```

You can assume that none of the integer values entered will be 0 (maybe you can convince the Bank of Mom and Dad to give you one of those special 0% interest loans).

Program Output

Output the monthly payment for the loan to the screen.

```
Your monthly car payment is: $xxx.xx
```

back to code wars VI

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

codewars.6

Program / Task Name:	**What Day Is It?**
Level:	**Novice**
Point Value:	**3**

Task Description

You've been tasked with creating a calendar for this year (2003). Instead of printing the entire calendar, you will prompt the user for a month and day. You will then display the month and day entered along with the day of the week on which that day occurs.

Program Input

Your program will prompt the user for:

- the number of the month, where 1=January, 2=February, 3=March, 4=April, 5=May, 6=June, 7=July, 8=August, 9=September, 10=October, 11=November, and 12=December
- the day of the week on which the month begins, where 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday

You can assume that only valid entries for the month and day of the month will be entered.

```
Enter the month: 3

Enter the day: 1
```

Program Output

Your program must display a month, day and day of the week in the following format:

```
March 1, 2003 is a Saturday
```

codewars.6

Program / Task Name:	**Corporate Name Changer**
Level:	**Novice**
Point Value:	**4**

Task Description

Congratulations! Your company, WidgetCo, has just merged with the much larger SuperMegaCorp, but don't worry, you've got a new assignment! WidgetCo has hundreds and hundreds of press releases and user manuals that mention the company by name. But since you're now part of SuperMegaCorp, all those documents need to be changed.

Your assignment is to write a program to read in a text file, find and replace every occurrence of "WidgetCo" with "SuperMegaCorp", and write out the updated document to another file. The file will be a simple ASCII text file containing up to 2 pages of text.

Program Input

The input file, prob04.in, contains a press release you need to modify:

```
WidgetCo, Houston, TX - April 1, 2003.  For immediate release.

WidgetCo announced today the immediate availability of the
UltraWidget 3000, the first all-titanium widget in a 26cm
diameter form factor.  The UltraWidget 3000 not only increases
cross-phase velocity, but also reduces temporal drag by 16%
(when compared to other widgets in its class).  WidgetCo Vice
President Bill Hottaire said this morning that "the WidgetCo
UltraWidget 3000 is the most advanced non-carbon-fiber widget
available".  The UltraWidget 3000 will be available from all
WidgetCo dealers by the end of September.  Pricing has not been
finalized, but is expected to be under $7000 per widget in
quantities of 1000 or more.
```

Program Output

Change each instance of WidgetCo to SuperMegaCorp and output the modified press release to prob04.out.

```
SuperMegaCorp, Houston, TX - April 1, 2003.  For immediate
release.

SuperMegaCorp announced today the immediate availability of the
UltraWidget 3000, the first all-titanium widget in a 26cm
diameter form factor.  The UltraWidget 3000 not only increases
cross-phase velocity, but also reduces temporal drag by 16%
(when compared to other widgets in its class).  SuperMegaCorp
Vice President Bill Hottaire said this morning that "the
SuperMegaCorp UltraWidget 3000 is the most advanced non-carbon-
fiber widget available".  The UltraWidget 3000 will be
available from all SuperMegaCorp dealers by the end of
September.  Pricing has not been finalized, but is expected to
be under $7000 per widget in quantities of 1000 or more.
```

codewars.6

Program / Task Name:	**Histogram**
Level:	**Novice**
Point Value:	**5**

Task Description

A histogram is a bar graph in which the area of each bar is proportional to an item's frequency of occurrence. Write a program to generate a histogram of the occurrences of the letters (A-Z) in a text file.

Program Input

A collection of strings. The input file prob05.in will not contain any blank lines. All letters will be uppercase (no need to check), and your program may ignore numbers, spaces, punctuation, etc. Lines will be no longer than 80 characters, and if a letter occurs more than 32 times the histogram should stop at 32.

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
THIS IS AN EXAMPLE OF HOW TO TEST YOUR
HISTOGRAM PROGRAM.  YOU
CAN USE THIS EXAMPLE.
```

Program Output

The program must display on screen a vertical histogram representing the occurrences of each letter of the alphabet:

```
                *
                *
     *          *
     *          *     *
*    *          *    **
*    *   *      *   ***
*    *   **    * *  ****
*    *   **    * ** ****
*    * ***   ***** ****  **
* * *****   ***** **** ***
**************************
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

codewars.6

Program / Task Name:	Checking the Sum
Level:	Novice/Advanced
Point Value:	5

Task Description

Before HP ships any computer, diagnostic software tests the components to ensure the machine works properly. One method for testing blocks of memory is called Checksum Verification. You will write a program to compute a checksum on a block of data.

- The checksum for this program is an eight bit unsigned value which, when added to the unsigned sum of the data values, zeros the sum. This value is called the two's compliment. For example, here is a four byte data block (all data values are in hexadecimal): 9E A6 58 80. The eight bit unsigned sum of the data values is 1C. The two's compliment of 1C is E4 (notice that the eight bit unsigned sum of 1C and E4 is zero).

A diagnostic test would compare the expected checksum against the computed checksum. If the values did not match, then the test would fail.

Program Input

The input file (prob06.in) consists of 256 eight bit unsigned hexadecimal numbers. There are 16 lines, and 16 values on each line, separated by spaces. The program should handle upper case and lower case letters.

```
54 61 73 6b 20 44 65 73 63 72 69 70 74 69 6f 6e
0d 0a 42 65 66 6f 72 65 20 68 70 20 73 68 69 70
73 20 61 6e 79 20 63 6f 6d 70 75 74 65 72 2c 20
64 69 61 67 6e 6f 73 74 69 63 20 73 6f 66 74 77
61 72 65 20 74 65 73 74 73 20 74 68 65 20 63 6f
6d 70 6f 6e 65 6e 74 73 20 74 6f 20 65 6e 73 75
72 65 20 74 68 65 20 6d 61 63 68 69 6e 65 20 77
6f 72 6b 73 20 70 72 6f 70 65 72 6c 79 2e 20 4f
6e 65 20 6d 65 74 68 6f 64 20 66 6f 72 20 74 65
73 74 69 6e 67 20 62 6c 6f 63 6b 73 20 6f 66 20
6d 65 6d 6f 72 79 20 69 73 20 63 61 6c 6c 65 64
20 43 68 65 63 6b 73 75 6d 20 56 65 72 69 66 69
63 61 74 69 6f 6e 2e 20 59 6f 75 20 77 69 6c 6c
20 77 72 69 74 65 20 61 20 70 72 6f 67 72 61 6d
20 74 6f 20 63 6f 6d 70 75 74 65 20 61 20 63 68
65 63 6b 73 75 6d 20 6f 6e 20 61 20 62 6c 6f 63
```

Program Output

The program must compute the eight bit unsigned sum of the data values and the checksum. The program must write these two numbers in hexadecimal to the output file (prob06.out). Please label the values as in the sample output given below:

```
unsigned sum: X5
checksum: XB
```

back to code wars VI

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

codewars.6

back to code wars VI

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

Program / Task Name:	**Music**
Level:	**Novice / Advanced**
Point Value:	**6**

Task Description

On a piano keyboard, if one begins at middle-C, then the sequence of half-steps from middle-C to B is listed in the following table. Since two different terminologies for naming the notes can be used, two different terminologies are shown, a “sharp” terminology and a “flat” terminology. All notes in the same row are identical.

Sharp terminology	Pronunciation	Flat terminology	Pronunciation
C		C	
C#	“C-sharp”	D	“D-flat”
D		D	
D#	“D-sharp”	E	“E-flat”
E		E	
F		F	
F#	“F-Sharp”	G	“G-flat”
G		G	
G#	“G-sharp”	A	“A-flat”
A		A	
A#	“A-sharp”	B	“B-flat”
B		B	

“#” and “ ” are known generically as an “accidentals.”

Immediately following the B is another identical sequence going from C to B through each of the notes above. Each of these sequences, when going from C to C, is called an octave. On an 88-key piano, this pattern repeats 4 times from middle-C to the highest C on the piano, and the highest note on the piano is a C. Thus, pattern stops at C, exactly 4 octaves above middle C.

A major chord is created by playing three notes together. The bottom note is the chord name. Thus if the bottom note is a C, then the chord is called C major. The other two notes are derived by moving 4 half steps up to get one note and then 3 half-steps up from there to get the other note. Thus, a C-major chord is C-E-G.

A minor chord is also created by playing three notes together. Again, the bottom note is the chord name, and if the bottom note is a C, for example, the chord is called C minor. To get the second note with a minor chord, one moves 3 half steps up; the third note is derived by moving 4 half steps up from there. Thus, a C-minor chord is C-D#-G (or C- E -G).

Write a program that reads the input file of individual notes and either the word, “major,” or “minor,” and outputs the three-note combination for the appropriate major or minor chord based on each note. Since the “ “ symbol is a complex Unicode character, your input file will use a lower-case “b” to represent flat; your output file should do the same. Your output must use the flat-names (e.g. “Eb”) whenever an accidental must be expressed if the input also used a flat name. If the input used neither flat nor sharp (e.g. “C”) or used a sharp name (e.g. “C#”), then all derived notes must use the sharp terminology whenever an accidental must be expressed. Use a hyphen (“-“) to separate notes in the output and place each output on its own line.

Program Input

The input file (prob07.in) consists of the individual notes. Exactly one space will separate the note specification from the word major or minor. A newline character (\n) will follow the word major or minor in each case. The number of lines of input is not specified for you. You must read until the end of file is reached. Except with regard to the “b” symbol that is used to indicate flat, your program must read the input data in a case-insensitive fashion.

```
C major
gb minor
Bb major
d# minor
```

Program Output

Output the three-note combination for the appropriate major or minor chord to a file (prob07.out):

```
C-E-G
Gb-A-Db
Bb-D-F
D#-F#-A#
```

codewars.6

back to code wars VI

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

Program / Task Name:	**MP3 Organizer**
Level:	**Novice / Advanced**
Point Value:	**8**

Task Description

You've decided to join the digital music revolution, and convert all of your CD's to digital media. You want your music to sound its best by using a high-bitrate MP3 format. Since you want to take your digital music with you everywhere, you decide to write a program to tell you how many MP3CD's (a 650MB CD-ROM full of MP3 song files) it will take to hold the collection once it has been converted.

Using a list of CD's and MP3's, calculate the number of MP3CD's required to hold your entire collection, using a given target bitrate. Your supported bitrates (used as keywords in the input file) are:

MP3-128 = 128 kilobit/second MP3
MP3-160 = 160 kilobit/second MP3
MP3-192 = 192 kilobit/second MP3
MP3-256 = 256 kilobit/second MP3
MP3-320 = 320 kilobit/second MP3
FLAC = 700 kilobit/second Lossless (assumes a 2:1 compression from CD Audio)
CD = 1411 kilobit/second CD Audio (WAV file)

For example, a 10 second CD Audio clip (WAV) uses 1.764MB of disk space (note that for our purposes, 1MB = 1000 kilobytes).

Program Input

The first line of the input file (prob08.in) contains the target bitrate (format) for encoding. Each line that follows will contain four fields about each CD or MP3 album in your collection, in the following format:

<artist name>, <album name>, <album length in minutes:seconds>, <format>

Sample Input file:

```
TARGET_BITRATE=MP3-256
Duran Duran, Rio, 45:44, CD
Flock of Seagulls, One Hit Wonder, 39:10, CD
Orbital, Orbital 2, 54:33, MP3-192
Beethoven, 9th Symphony, 73:52, CD
Britney Spears, Blah Blah Blah, 33:33, 128
Metallica, Please Give Us Money, 46:58, MP3-256
The Barking Dogs, A Barking Dog Christmas, 68:04, CD
```

Note that if a particular album is already encoded at a lower bitrate than the target bitrate, you will just store the album as is, because converting it to a higher-rate format would be a waste of disk space. For example, Britney Spears' "Blah Blah Blah" is shown as being in the MP3-128 format, and the target bitrate is MP3-256, so that album would just be stored as an MP3-128. But the Matrix Reloaded Soundtrack is in the CD format, and therefore would be encoded in the MP3-256 format.

Program Output

Your output must be written to a file (prob08.out) and contain an album listing for each MP3CD, with

a header that includes the CD number (start at #1) separating each MP3CD. The album listing must show the artist, album name, time, and file size in megabytes (in whole MB, round up). And finally, the total amount of space used on the MP3CD must be shown (in whole MB, round up).

To calculate the number of MP3 CD's correctly, fill each MP3CD in the order the albums are shown in the input file. If there isn't enough space left to hold the next album, start on the next CD.

Sample output file:

```
MP3 CD #1
-------------------
Duran Duran, Rio, 45:44, 88MB
Flock of Seagulls, One Hit Wonder, 39:10, 76MB
Orbital, Orbital 2, 54:33, 79MB
Beethoven, 9th Symphony, 73:52, 142MB
Britney Spears, Blah Blah Blah, 33:33, 33MB
Metallica, Please Give Us Money, 46:58, 91MB
The Barking Dogs, A Barking Dog Christmas, 68:04, 131MB

Total: 637MB
```

codewars.6

back to code wars VI

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

Program / Task Name:	**What's My Score?**
Level:	**Novice / Advanced**
Point Value:	**10**

Task Description

A single bowling game consists of ten frames. The object in each frame is to roll a ball at ten bowling pins arranged in an equilateral triangle and to knock down as many pins as possible.

For each frame, a bowler is allowed a maximum of two rolls to knock down all ten pins. If the bowler knocks them all down on the first attempt, the frame is scored as a strike. If the bowler does not knock them down on the first attempt in the frame the bowler is allowed a second attempt to knock down the remaining pins. If the bowler succeeds in knocking the rest of the pins down in the second attempt, the frame is scored as a spare.

The score for a bowling game consists of sum of the scores for each frame. The score for each frame is the total number of pins knocked down in the frame, plus bonuses for strikes and spares. In particular, if a bowler scores a strike in a particular frame, the score for that frame is ten plus the sum of the next two rolls. If a bowler scores a spare in a particular frame, the score for that frame is ten plus the score of the next roll. If a bowler scores a strike in the tenth (final) frame, the bowler is allowed two more rolls. Similarly, a bowler scoring a spare in the tenth frame is allowed one more roll.

The maximum possible score in a game of bowling (strikes in all ten frames plus two extra strikes for the tenth frame strike) is 300.

Program Input

The input file (prob09.in) consists of a sequence of bowling game scores. Each line will contain the scores for a single game. The score for a single roll will be represented by a single character -- either a number indicating the number of pins knocked down, a '/' for a spare or a capital 'X' for a strike:

```
9/X9/X9/X9/X9/9/9
XXXXXXXXXXXX
909090909090909090
```

Program Output

Output the actual score sheet for each game, complete with pin count and cumulative score for each frame as follows:

```
 Game       1    2    3    4    5    6    7    8    9     10
-----------------------------------------------------------------
|        | 9|/|  |X| 9|/|  |X| 9|/|  |X| 9|/|  |X| 9|/| 9|/|9|
|--------|----|----|----|----|----|----|----|----|----|------|
| Game 1 |  20|  40|  60|  80| 100| 120| 140| 160| 179|   198|
-----------------------------------------------------------------
|        |  |X|  |X|  |X|  |X|  |X|  |X|  |X|  |X|  |X| X|X|X|
|--------|----|----|----|----|----|----|----|----|----|------|
| Game 2 |  30|  60|  90| 120| 150| 180| 210| 240| 270|   300|
-----------------------------------------------------------------
|        | 9|0| 9|0| 9|0| 9|0| 9|0| 9|0| 9|0| 9|0| 9|0| 9|0|  |
|--------|----|----|----|----|----|----|----|----|----|------|
| Game 3 |   9|  18|  27|  36|  45|  54|  63|  72|  81|    90|
-----------------------------------------------------------------
```

codewars.6

back to code wars VI

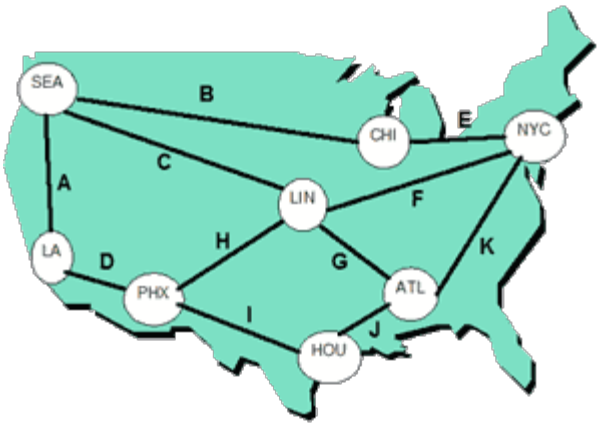
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	Data Transmission
Level:	Novice / Advanced
Point Value:	10

Task Description

In this problem your task is to find the quickest path for information to flow from a source city to a destination city. The transmission can start from any city and end up at any city. Your task is to always find the fastest based on the transmission times between cities.

Using the map at right, you will be given a set of values for each of the numbered links, followed by a start city and destination city. Your challenge is to always find the quickest path for the data and display the cities as the data travels through them. Each of the lettered links will be a transmission time. (A = 1 second, B = 2 seconds, etc….)



Program Input

Your program will read in the link times from “A” to “K” and a series of source and destination cities from prob10.in. The first line will be the link time for link “A”, and the 11th line will be the link time for link “K”. The lines following those will be source city followed by a space and the destination city.

```
5
10
2
6
4
1
1
3
2
2
5
SEA HOU
NYC LA
CHI SEA
PHX HOU
```

Program Output

Output to the screen the trip title, the cities traveled through, including start and destination and the total transmission time (from start to destination).

```
For SEA to HOU: SEA, LIN, ATL, HOU in 5 seconds
For NYC to LA:  NYC, LIN, PHX, LA in 10 seconds
For CHI to SEA:  CHI, NYC, LIN, SEA  in 7 seconds
For PHX to HOU:  PHX, HOU in 2 seconds
```

codewars.6

Program / Task Name:	**Create a Monthly Planner**
Level:	**Advanced**
Point Value:	**11**

Task Description

You've been tasked with creating a calendar for a particular month of the year. You will also create a calendar for the previous month and the next month.

Program Input

Your program will prompt the user for:

- the number of the month, where 1=January, 2=February, 3=March, 4=April, 5=May, 6=June, 7=July, 8=August, 9=September, 10=October, 11=November, and 12=December
- the day of the week on which the month begins, where 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday

You can assume:

- the year in question IS NOT a leap year,
- only valid values (for the month and day) will be entered, and
- the months of January and December will not be entered as the month (values 1 and 12)

```
Enter the number of the month for the calendar: 3
Enter the day on which the month begins: 7
```

Program Output

Your program must display a 3 month calendar containing the month specified, the previous month, and the next month as follows:

```
February                  March                     April
Su  M Tu  W Th  F Sa      Su  M Tu  W Th  F Sa      Su  M Tu  W Th  F Sa
                   1                         1                 1  2  3  4  5
 2  3  4  5  6  7  8       2  3  4  5  6  7  8       6  7  8  9 10 11 12
 9 10 11 12 13 14 15       9 10 11 12 13 14 15      13 14 15 16 17 18 19
16 17 18 19 20 21 22      16 17 18 19 20 21 22      20 21 22 23 24 25 26
23 24 25 26 27 28         23 24 25 26 27 28 29      27 28 29 30
                          30 31
```

codewars.6

Program / Task Name:	**HP Calculator**
Level:	**Advanced**
Point Value:	**13**

Task Description

Some hp calculators, such as the 12c, use an entry system logic called RPN. Sometimes this is called the 'enter method' because the calculator has no EQUAL key, but uses an ENTER key instead. Many scientists and engineers prefer this entry system logic. You will write a program to process RPN expressions.

In summary, RPN does not use parentheses, and operators (like plus and minus) follow their operands. A few examples are shown below:

Algebraic Expression	RPN
3 + 5	3 5 +
6 * (7 + 8)	6 7 8 + *
(9 + 5) * (7 + 8)	9 5 + 7 8 + *
((4 * (2 – 3)) / (5 – 6)	4 2 3 – * 5 6 – /

Program Input

Each line of the input file (prob12.in) contains a series of numbers and operators. The program must evaluate these lines as RPN expressions. All tokens are separated by one or more spaces. Numbers may be negative. Valid operators are +(addition), -(subtraction), *(multiplication) and /(division). The last line of the file contains only the word END:

```
16 3 7 + *
4 32.125 13 – * 20 +
5 –3 * 4 2 / 6 3 1 – / + +
END
```

Program Output

The program will print the computed result of each RPN expression on a separate line.

```
160
96.5
-10
```

codewars.6

Program / Task Name:	**Magic Square**
Level:	**Advanced**
Point Value:	**13**

Task Description

The Magic Square is a classic math puzzle. Given a semi empty 4x4 matrix, you must determine which values to place in each element to match the row, column, and diagonal sums.

```
                .
|  |  |13|  |  67
| 3|  |  |  |  37
|  |  |  |11|  34
|  |15|  |  |  69

 52 48 31 76   38
```

Program Input

Each line in the file prob13.in gives the values for each row in the magic square. Blank values that need to be filled in are represented by XX, like this:

```
XX XX 13 XX 67
 3 XX XX XX 37
XX XX XX 11 34
XX 15 XX XX 69
52 48 31 76 38
```

Program Output

Determine the missing values and print the entire magic square to the output file, prob13.out. You may assume all values in the square, including the sums, are within the range 0 to 99, inclusive.

```
 9 27 13 18 67
 3  1 12 21 37
16  5  2 11 34
24 15  4 26 69
52 48 31 76 38
```

back to code wars VI

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

codewars.6

Program / Task Name:	**Assembler**
Level:	**Advanced**
Point Value:	**13**

Task Description

A simple computer has been designed to have the following general registers that are each 16-bits wide: AX, BX, CX, and DX.

This same computer has the following simple operations that it can perform on any of the registers:

- MOV reg1, reg2 - moves value of reg2 into reg1.
- MOV reg1, <value> - moves numeric value specified into reg1.
- ADD reg1, reg2 - adds value of reg2 to reg1 and stores the sum in reg1.
- ADD reg1, <value> - adds the numeric value specified to reg1 and stores this sum in reg1.
- SUB reg2, reg1 - subtracts the value in reg2 from the reg1 and stores this in reg1.
- SUB reg1, <value> - subtracts the numeric value specified from reg1 and stores this in reg1.
- SHOW reg1 - displays value in reg1 (in decimal) followed by a CR/LF.
- HLT - signifies the end of the program.

Note: regx can be any of the registers (AX, BX, CX, or DX) and <value> is any whole decimal number 0 – 65535. Assume that only a comma separates the destination and source fields.

Your task is to write a code interpreter that will execute a given program to completion.

Program Input

Your program will be given an input file (prob14.in) with one operation per line. It should execute all instructions until it reaches a “HLT”, and then exit. A sample input file is shown below:

```
MOV AX,3
MOV BX,2000
ADD BX,AX
SHOW AX
SHOW BX
HLT
```

Program Output

Write the output to prob14.out as follows:

```
3
2003
```

back to code wars VI

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

codewars.6

back to code wars VI

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

Program / Task Name:	Reverse Crossword
Level:	Advanced
Point Value:	17

Task Description

You will be given an n x m ASCII representation of a solved crossword puzzle. Each single character represents a square of the crossword puzzle. Blank spaces will be denoted by an asterisk (*). Your job is to construct the list of words that make up the crossword answers. Words must consist of at least two (2) letters and can run across from left to right or they may run down from top to bottom. Each word should be printed along with its corresponding number. A word's number refers to the number of the square that contains the first letter of the word. Squares should be numbered from left-to-right first, then top-to-bottom starting with number 1. Only squares that start a word get a number.

Program Input

The first line of input file (prob15.in) will be two integers n and m separated by a single space denoting the number of rows and columns, respectively, in the crossword puzzle. The next n lines contain m characters each and represent the solved crossword puzzle:

```
15 15
*t*s*o*e*f*g*m*
serendip*atonal
*n*r*y*s*g*i*g*
derbyshire*nero
*t***s*l*n*g*i*
ashore*obdurate
***d*u*n***o*t*
yiddish*accused
*a*f***s*o*n***
amnesiac*nodose
*b*l*b*h*t***t*
curl*icebreaker
*s*o*d*m*a*s*a*
peewee*escapade
*s*s*m*s*t*s*y*
```

Program Output

Your program should output the word "Across" on it's own line followed by a line for each "across" number and word to prob15.out. You should repeat the same pattern for the "down" words:

```
Across
8.      serendip
9.      atonal
10.     derbyshire
11.     nero
12.     ashore
14.     obdurate
15.     yiddish
17.     accused
20.     amnesiac
22.     nodose
24.     curl
25.     icebreaker
27.     peewee
28.     escapade
Down
```

```
1.      tenets
2.      serb
3.      odysseus
4.      epsilon
5.      fagend
6.      goinground
7.      magritte
13.     oddfellows
16.     iambuses
18.     contract
19.     schemes
21.     ibidem
23.     steady
26.     asps
```