Beginner Problems

Problem 1: Unit Conversion

100 points
Filename: prob01 *(e.g. prob01.c, prob01.cpp, prob01.java, prob01.py2, prob01.py3)*

Description
We need to calculate the weight of various containers in pounds (LB). One kilogram (KG) equals 2.2 pounds (LB).

Each line of input will contain number of liters (L) per container and density of the contents (KG/L).

Multiply the number of liters times the weight per liter to determine total weight in kilograms (KG). Then convert the total weight from kilograms (KG) to pounds (LB) by multiplying kilograms (KG) by 2.2.

Liters x KG/L = Total KG, then Total KG x 2.2 = Total LB

Example:

937.50 Liters x 0.866 KG/L = 811.875 KG, then 811.875 KG x 2.2 LB/KG = 1786.125 LB

Output the weight in Pounds (LB) for each container. Round your result to the nearest hundredth of a pound (two decimal places). Always display two decimal places using natural rounding (e.g. 1786.125 = 1786.13).

Sample Input

```
1000 1
2000 0.91
937.50 0.866
```

Sample Output

```
2200.00
4004.00
1786.13
```

Learn More
Liquid density varies by type of liquid. Water weighs 1 kilogram per liter, while olive oil only weighs 0.91 kilograms per liter, and crude oil weighs even less at 0.88 kilograms per liter. The low density of crude oil causes it to float on top of water when an oil spill occurs.

Problem 2: Rounding

100 points
Filename: prob02 *(e.g. prob02.c, prob02.cpp, prob02.java, prob02.py2, prob02.py3)*

Description
Given the size of 5 files in megabytes, we will calculate the combined disk usage to the nearest half gigabyte (GiB). One gigabyte (GiB) equals 1024 megabytes (MiB).

Each line of input will contain five file sizes in megabytes (MiB).

Sum the 5 file sizes, then convert the total file sizes from megabytes (MiB) to gigabytes (GiB) by dividing by 1024, then round to the nearest half gigabyte (GiB).

Example:

250 MiB + 500 MiB + 750 MiB + 1500 MiB + 2250 MiB = 5250 MiB
5250 MiB / 1024 = 5.13 GiB = 5.0 GiB

Output the total size in gigabytes (GiB). Round your result to the nearest half gigabyte (GiB). Always display one decimal place (e.g. 5.13 = 5.0).

Sample Input

```
250 500 750 1500 2250
100 200 300 400 500
150 350 550 750 950
```

Sample Output

```
5.0
1.5
2.5
```

Tip
A simple way to round a number to the nearest half would be to multiply the amount by two, round to nearest whole number, then divide by two (e.g. 1.25 x 2 = 2.5, round(2.5) = 3, 3 / 2 = 1.5).

Learn More
The megabyte is commonly used to measure either $1{,}024^2$ bytes (MiB) or $1{,}000^2$ bytes (MB). When using the **binary system**, 1 MiB = 1,048,576 bytes. This traditional definition is becoming less common, in favor of the decimal system. When using the **decimal system**, 1 MB = 1,000,000 bytes. This definition is used in networking contexts and most storage media.

Prepared by Jason Klein
https://en.wikipedia.org/wiki/Megabyte
https://en.wikipedia.org/wiki/File_size

Problem 3: Hosting Costs

100 points
Filename: prob03 *(e.g. prob03.c, prob03.cpp, prob03.java, prob03.py2, prob03.py3)*

Description
We are helping someone setup cloud services for a website. They must pay for the resources they use, such as server time, data stored, and data downloaded (aka bandwidth). Given an estimated number of server hours for the month, server cost per hour, number of gigabytes storage, cost per gigabyte of storage, number of gigabytes bandwidth, and cost per gigabyte of bandwidth, calculate their total hosting costs for the month.

Begin by calculating server costs:
    2,880 hours x $0.0052/hour = $14.976 server cost
Then calculate storage costs:
    100GiB x $0.023/GiB = $2.30 storage costs
Then calculate bandwidth costs:
    100GiB x $0.09/GiB = $9.00 bandwidth costs
Finally, add server, storage, and bandwidth costs to determine total hosting costs for the month:
    $14.976 + $2.30 + $9.00 = $26.276 total hosting costs

Each line of input will contain the number of server hours for the month, server cost per hour, number of gigabytes storage, cost per gigabyte of storage, number of gigabytes bandwidth, and cost per gigabyte of bandwidth.

Output the total hosting costs for each configuration. Round to two decimal places. Always display two decimal places (e.g. 47.002 = 47.00).

Sample Input

```
2880 0.0052 100 0.023 100 0.09
1440 0.0208 500 0.0125 120 0.09
720 0.0052 5 0.10 5 0.09
```

Sample Output

```
26.28
47.00
4.69
```

Hints
You may assume all hours, gigabytes, and costs will be greater than or equal to zero.

Learn More
The costs above are based on actual cloud service costs. If you are hosting media files (e.g. photos), use an object store (e.g. $0.023/GB) instead of a server disk ($0.10/GB) to keep storage costs down.

Problem 4: Tribonacci Sequence

100 points
Filename: prob04 *(e.g. prob04.c, prob04.cpp, prob04.java, prob04.py2, prob04.py3)*

Description
Mathematicians define the Fibonacci Sequence to be an integer sequence where every number—after the first two—is the sum of the two preceding ones.

```
Fibonacci Sequence:  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
```

A similar integer sequence explored in computer science classes is the Tribonacci Sequence. The Tribonacci sequence is an integer sequence where every nth number in the sequence—after the first three—is the sum of the three preceding ones as shown below:

$n \rightarrow$	0	1	2	3	4	5	6	7	8	9	10	11	...
Tribonacci number:	1	1	2	4	7	13	24	44	81	149	274	504	...

In other words, the 6th Tribonacci number (24) is the sum of 13, 7, and 4.

The input contains one or more lines. Each line contains one positive integer representing n such that $0 \leq n \leq 50$.

For each value of n, produce the Tribonacci number.

Sample Input

```
2
10
6
```

Sample Output

```
2
274
24
```

https://brilliant.org/wiki/tribonacci-sequence/
Provided by Charles Moss. Edited by Jason Klein.

Intermediate Problems

Problem 5: Word Folding

150 points
Filename: prob05 *(e.g. prob05.c, prob05.cpp, prob05.java, prob05.py2, prob05.py3)*

Description
Each line of input contains a single word. The length of the word will be at least 1 character and no more than 50 characters. Reorder the letters of each word and output the new word. The first letter output should be the first letter of the word, then the last letter of the word, then the second letter of the word, then the second to the last letter of the word, etc. until all letters have been output.

Each line of input will contain a word that needs to be folded.

Each line of output should contain the new word.

Sample Input

```
HELLO
WORLD
SOFTWARE
```

Sample Output

```
HOELL
WDOLR
SEORFATW
```

Tips
The original word will only contain uppercase letters. The word will NOT contain any numbers, spaces, or other special characters. Do not alter the capitalization of any of the letters in the word.

Looking for more like this? Visit https://rosettacode.org/wiki/Category:Simple
Provided by Charles Moss. Edited by Jason Klein.

Problem 6: Check Digit

150 points
Filename: prob06 *(e.g. prob06.c, prob06.cpp, prob06.java, prob06.py2, prob06.py3)*

Description
Given the first 9 digits (e.g. 012345678) of a 10-digit ISBN number, calculate the check digit and output the 10-digit ISBN number. Assuming the first 9 digits of the ISBN are "$d_1\,d_2\,d_3\,d_4\,d_5\,d_6\,d_7\,d_8\,d_9$", the check digit is computed with this formula:

$$\left[\sum_{i=1}^{9} i(d_i)\right] \bmod 11$$

This means the check digit is calculated as the sum of the products of each digit times its position in the list with the sum modulo by 11 to determine the remainder. The value of the remainder becomes the last digit in the ISBN. Here are two steps you can use to reproduce the formula above:

Step 1: (1 x d_1) + (2 x d_2) + (3 x d_3) + (4 x d_4) + (5 x d_5) + (6 x d_6) + (7 x d_7) + (8 x d_8) + (9 x d_9) = Total
Example: (1 x 0) + (2 x 3) + (3 x 0) + (4 x 6) + (5 x 4) + (6 x 0) + (7 x 6) + (8 x 1) + (9 x 5) = 145

Step 2: The checksum is the total mod 11 (e.g. 145 % 11 = 2).

Each line of input will contain a 9-digit integer.

The output should contain the full 10-digit ISBN number. Be sure to pad the beginning of the number with zeroes as necessary. A checksum digit value of 10 should be represented as "X".

Sample Input

```
030640615
039480001
039480002
```

Sample Output

```
0306406152
039480001X
0394800028
```

Learn More
The International Standard Book Number (ISBN) began associating a 10-digit code with a published book in 1970. The ISBN includes sections that identify the origin country, area, language, publisher, a number assigned by the publisher to the book in question, and a single check digit.

https://en.wikipedia.org/wiki/International_Standard_Book_Number
https://en.wikipedia.org/wiki/Modulo_operation
Provided by Charles Moss. Edited by Jason Klein.

Problem 7: Caesar Cipher

150 points
Filename: prob07 *(e.g. prob07.c, prob07.cpp, prob07.java, prob07.py2, prob07.py3)*

Description
Julius Caesar used what is now considered a very simple encryption system. He shifted each letter in a message 3 letters to the right, wrapping around from the right end to the left end if needed, as shown below:

Original Letter:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Encrypted Letter:

```
DEFGHIJKLMNOPQRSTUVWXYZABC
```

Accept a single word as input. The length of the word will be at least 1 character and no more than 50 characters. Shift each letter three steps to the right, wrapping around if necessary, and output the resulting word.

Each line of input will contain a word that needs to be encrypted.

Sample Input

```
HELLO
WORLD
CAESAR
CIPHER
FREQUENCY
DISTRIBUTION
```

Sample Output

```
KHOOR
ZRUOG
FDHVDU
FLSKHU
IUHTXHQFB
GLVWULEXWLRQ
```

Tips
The original word will only contain uppercase letters. The word will NOT contain any numbers, spaces, or other special characters. Do not alter the capitalization of any of the letters in the word.

https://en.wikipedia.org/wiki/Caesar_cipher
Provided by Charles Moss. Edited by Jason Klein.

Problem 8: Duplicate Words

150 points
Filename: prob08 *(e.g. prob08.c, prob08.cpp, prob08.java, prob08.py2, prob08.py3)*

Description
There is a game in which you try not to repeat a word while your opponent tries to see if you have repeated one.

> "THE RAIN IN SPAIN" does not repeat any words.
> "IN THE RAIN AND THE SNOW" repeats THE.
> "THE RAIN IN SPAIN IN THE PLAIN" repeats THE and IN.

Write a program to test a phrase and determine if any words repeat.

Input
Input is a line containing words separated by single spaces, where a word consists of one or more uppercase letters. A line contains no more than 80 characters.

Output
The output is "FALSE" if no word is repeated, and "TRUE" if one or more words are repeated.

Sample Input

```
THE RAIN IN SPAIN
```

Sample Output

```
FALSE
```

Sample Input

```
IN THE RAIN AND THE SNOW
```

Sample Output

```
TRUE
```

Sample Input

```
THE RAIN IN SPAIN IN THE PLAIN
```

Sample Output

```
TRUE
```

https://en.wikipedia.org/wiki/The_Rain_in_Spain
Provided by Charles Moss. Edited by Jason Klei

Advanced Problems

Problem 9: Cubic Squares

200 points
Filename: prob09 *(e.g. prob09.c, prob09.cpp, prob09.java, prob09.py2, prob09.py3)*

Description
A number is said to be a perfect square if it is the product of an integer multiplied by itself (e.g. 25 is a perfect square, because $5^2 = 5 \times 5 = 25$). A number is said to be a perfect cube if it is the product of an integer multiplied by the square of that integer (e.g. 125 is perfect cube, because $5^3 = 5 \times 5 \times 5 = 125$). Some numbers are actually both a perfect square and a perfect cube (e.g. 64 is both, because $8^2 = 64$ and $4^3 = 64$).

Accept a series of five non-negative integers as input. The numbers may or may not be entered in sorted order, so do not assume that they are sorted, and do not alter their order. Evaluate each one to determine if it is a perfect square, a perfect cube, both, or neither.

Output the evaluation of each of the five integers as “NEITHER”, “PERFECT SQUARE”, “PERFECT CUBE”, or “BOTH”.

Sample Input

```
1
17
25
64
125
```

Sample Output

```
BOTH
NEITHER
PERFECT SQUARE
BOTH
PERFECT CUBE
```

Learn More
In mathematics, a **square** number, or a perfect square, is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, 9 is a square number, since it can be written as 3 x 3.

A **cube** number, or a perfect cube, or sometimes just a cube, is a number which is the cube of an integer. For example, 27 is a cube number, since it can be written as 3 x 3 x 3.

https://en.wikipedia.org/wiki/Square_number
https://en.wikipedia.org/wiki/Cube_(algebra)
Provided by Charles Moss. Edited by Jason Klein.

Problem 10: Binary Ones

200 points
Filename: prob10 *(e.g. prob10.c, prob10.cpp, prob10.java, prob10.py2, prob10.py3)*

Description
Any integer can be represented in Base-2 (also known as Binary). There are some times when it may be useful to know how many of the bits in the Base-2 (Binary) representation of a number are ones.

```
Base 10    Base 2
0          0000 0000
1          0000 0001
2          0000 0010
3          0000 0011
        ...
126        0111 1110
127        0111 1111
128        1000 0000
```

Accept a series of five non-negative integers as input. The numbers may or may not be entered in sorted order, so do not assume that they are sorted, and do not alter their order.

Output the number of ones in the Base-2 (Binary) representation of each of the five integers.

Sample Input

```
1
3
126
127
128
```

Sample Output

```
1
2
6
7
1
```

Tips
Be sure to test input numbers larger than 255.

https://en.wikipedia.org/wiki/Binary_number
Provided by Charles Moss. Edited by Jason Klein.

Problem 11: Slugging Percentage

200 points
Filename: prob11 *(e.g. prob11.c, prob11.cpp, prob11.java, prob11.py2, prob11.py3)*

Description
In baseball, a player's batting average is calculated by dividing the total number of base hits by the total number of official at-bats. One limitation of using the batting average to evaluate players is that it treats all hits equally, rather than taking into account doubles, triples, or home runs. For this reason, analysts often prefer to consider what is known as the **slugging percentage**, which distinguishes between different hit outcomes. To calculate the **slugging percentage**, the total number of bases from all hits is divided by the total number of times at bat that did not result in walks.

More specifically, an at-bat can earn 0, 1, 2, 3, or 4 bases. (These are referred to as official at-bats.) Furthermore, some at-bats, such as those that result in a base-on-balls (i.e., a "walk") are not considered in either the player's batting average or slugging percentage.

For example, if a player hits a triple (3 bases), strikes out (0 bases), and hits a double (2 bases), their slugging percentage would be (3+0+2) / 3 ≈ 1.6667. If a player hits a single (1 base), walks, and hits a home run (4 bases), the slugging percentage would be (1+4) / 2 = 2.5. Notice that in this case, the denominator is two, not three, because the walk does not count towards the slugging percentage.

Input
Each line of input contains n integers, separated by spaces, each describing one of those at-bats. Strike-outs, singles, doubles, triples, and home runs are represented as 0, 1, 2, 3, and 4, respectively. Walks are represented as −1. You may assume that there will always be at least one official at-bat (i.e., at least one at-bat will not be a walk).

Output
For each line of input, display the player's slugging percentage as a real number, accurate to within an absolute or relative error of 10^{-3}. Round the value to **four** decimal places.

Sample Input

```
3 0 2
1 -1 4
-1 -1 0 0 0 0 0 0 1
```

Sample Output

```
1.6667
2.5
0.1429
```

https://en.wikipedia.org/wiki/Sabermetrics
https://en.wikipedia.org/wiki/Slugging_percentage
Provided by Charles Moss. Edited by Jason Klein.

Problem 12: Sator Square

200 points
Filename: prob12 *(e.g. prob12.c, prob12.cpp, prob12.java, prob12.py2, prob12.py3)*

Description
Accept a group of five 5-letter words as input and load them into a two-dimensional array. Check to see if the first row has the same spelling as the first column, the reverse of the last column, and the reverse of the last row. Then check to see if the second row is the same as the second column, the reverse of the second-to-last column, and the reverse of the second-to-last row. Then check to see if the third row is a palindrome and is the same as the third column.

If all these conditions are true, then print ”TRUE” (as a 4-letter string), or else print ”FALSE” (as a 5-letter string).

Sample Input

```
SATOR
AREPO
TENET
OPERA
ROTAS
```

Sample Output

```
TRUE
```

Sample Input

```
ABCDE
FGHIJ
KLMNO
PQRST
UVWXY
```

Sample Output

```
FALSE
```

Sample Input

```
ABCDE
BCDED
CDEDC
DEDCB
EDCBA
```

Sample Output

```
TRUE
```

Learn More
The Sator Square is a square 2D palindrome, which is when a square text admits four symmetries: identity, two diagonal reflections, and 180-degree rotation. As can be seen, the text may be read top-to-bottom, bottom-to-top, left-to-right, or right-to-left; and it may be rotated 180 degrees and still be read in all those ways.

The Sator Square is the earliest datable 2D palindrome. It was found in the ruins of Pompeii, at Herculaneum, a city buried in the ash of Mount Vesuvius in 79 AD. It consists of a sentence written in Latin: "Sator Arepo Tenet Opera Rotas." Its translation has been the subject of speculation with no clear consensus.

https://en.wikipedia.org/wiki/Sator_Square
Provided by Charles Moss. Edited by Jason Klein

Index