DKC^3 2019 – Programming Problems

Important Reminders:

- Time remaining will be announced at 30 min., 5 min. and 1 min. left in the session.

- Assume leading zeros are used in all decimal notations. (ex: 0.167)

- Example input and output files are available in the **C:\DKC3** folder for each problem. These may be used to test your programs.

- Each program must read from its respective input file in the **C:\DKC3** folder and generate an output file in that same folder. The names of these files must match what is specified for each problem in this document. These output files will be retrieved electronically and scored by the judges at the end of the session.

- Do **NOT** store any of your code in the **C:\DKC3** folder. Anything stored there will be overwritten.

- At the end of the session, judges will overwrite the example input files with the official input files for each problem. Each file will contain 10 test cases. One member of each team will be asked by a competition organizer to run all completed programs. Programs may be recompiled at this point prior to being run, but no changes can be made to source code.

- Each program must complete execution within **two** minutes.

- After running your programs, be sure to close out of all IDEs.

DKC^3 2019 – Programming Problems

1. Goldbach Partitions (20 points)

Goldbach's Conjecture states that every even integer greater than 2 can be expressed as the sum of two prime numbers. Although many have tried, this conjecture has never been proven; luckily, your job is not to prove it but to verify it for some values.

Your task is this: given an even integer N, find two prime numbers *a* and *b* such that *a* + *b* = N. If there is more than one way to select *a* and *b* then the correct partition should be chosen to minimize the product *a***b* (for example 18 can be expressed as 11+7 or 13+5, but since 13*5=65 < 11*7=77, the correct answer is 13+5).

Input
The input will consist of ten test cases, one per line. Each test case will be a single positive even integer N.

Output
The output for each test case should be a single line with the two prime numbers *a* and *b*. To ensure your output matches the solution, print the smaller of the two numbers first.

Input File: C:\DKC3\GoldbachIn.txt
Output File: C:\DKC3\GoldbachOut.txt

Examples:

Input:
36
6

Output:
5 31
3 3

2. Acceptable Passwords (15 points)

Write a program that reads a list of user passwords and prints the acceptable passwords to the output file. Each password will be separated by a space. Acceptable passwords may consist of letters, numbers, and special characters (defined below) and must meet the following criteria:

- Must be at least 8 characters long.
- Must not be longer than 32 characters.
- Must contain at least one uppercase letter (**A-Z**), one lowercase letter (**a-z**), one number (**0-9**), and one special character (**~ ! @ # $ % ^ & * ( ) - _ + = ? /**)
- If the password contains a non-alphanumeric character that is not included in the special characters above, it is considered unacceptable.

Input
The input will consist of ten test cases, one per line. Each test case will contain several user passwords, with each password separated by a space.

Output
The output for each test case should be a single line containing all the acceptable passwords found, with each acceptable password separated by a space. If there are no acceptable passwords found, print out "No acceptable passwords found!" followed by a space and the number of total passwords processed in that test case.

Input File: C:\DKC3\PasswordIn.txt
Output File: C:\DKC3\PasswordOut.txt

Examples:

Input:
DKC3Competition2019 Dkc3? (CodingRules!)
password P@ssword1 Pass.word1! Thisisap@ssword0000 Y=mx+b lengthXwidth=25

Output:
No acceptable passwords found! 3
P@ssword1 Thisisap@ssword0000 lengthXwidth=25

DKC^3 2019 – Programming Problems

3. Magic Number (15 points)

In baseball, among other sports, a magic number is a number used to indicate how close a front-running team is to clinching a division title and/or a playoff spot. It represents the total of additional wins by the front-running team or additional losses (or any combination thereof) by the rival teams after which it is mathematically impossible for the rival teams to capture the title in the remaining number of games (assuming some highly unlikely occurrence such as disqualification or expulsion from the competition or retroactive forfeiture of games does not occur).

The magic number is calculated as G + 1 − WA − LB, where
G is the total number of games in the season
WA is the number of wins that Team A has in the season
LB is the number of losses that Team B has in the season

For example, in Major League Baseball there are 162 games in a season. Suppose the top of the division standings late in the season are as follows:

Team	Wins	Losses
A	96	58
B	93	62

Then the magic number for Team B to be eliminated is 162 + 1 - 96 - 62 = 5.

Input/Output information on the next page.

Input
Read an input file containing the number of games in a season, followed by the name and win/loss record of each team. There will be ten test cases, and each test case will consist of multiple lines. One line for the number of games in the season followed by one line for each team's win/loss record. The teams will be listed in no particular order. The number of teams in each test case is unknown. Each test case (including the last) will be separated by a line containing a single hyphen.

Output
The output file should contain one line per test case. Each line will contain the first place team's name, followed by a space and magic number. If the magic number is <= 0, then the team has clinched a playoff spot. In this case, replace the magic number with "CLINCHED!" (without the quotes).

Input File: C:\DKC3\MagicIn.txt
Output File: C:\DKC3\MagicOut.txt

Examples:

Input:

```
137404787538
Koala 115019153641 3648378352
Otter 21925419785 96742112208
Platypus 94343645872 24323886121
-
531869027800
Coyote 176972420784 18815053748
Scorpion 47870179701 147917294831
Turkey 136557105767 59230368765
Lynx 8230943973 187556530559
Anteater 39528909944 156258564588
Stingray 179024007350 16763467182
Yak 52240116555 143547357977
Grouse 89537375611 106250098921
-
```

Output:

```
Koala CLINCHED!
Stingray 334029966703
```

4. Factorial Number (25 points)

Write a program that will read in a valid integer *n* from 0 to 20000 and will calculate the value of the rightmost non-zero digit of *n*! where *n*! is the product of all the integers 1 to *n* inclusive. Output will be a single digit integer from 1 to 9 representing the rightmost non-zero digit.

Input
The input will consist of ten test cases, each on its own line. Each test case will contain an integer *n.*

Output
For each test case, output an integer representing the rightmost non-zero digit of *n*!

Input File: D:\DKC3\FactorIn.txt
Output File: D:\DKC3\FactorOut.txt

Examples:

Input:
4
10

Output:
4
8

5. Profit Margin (5 points)

Billy Batch has been hired by a paint store to help them determine which color paint has the highest margin, as the store wants to maximize profits. He is given a list of 6 paints, the number of gallons sold, the total sale, and the cost per gallon. He would like you to help him order the paints by highest margin to lowest.

Note: Margin is calculated as (TOTAL_SALE – TOTAL_COST) / TOTAL_SALE.

Input
There will be ten test cases, each separated by an asterisk (*). Each test case will consist of 6 lines, with each line containing the color of the paint, the number of gallons sold, the total sale, and the cost per gallon. Each of these values will be whole numbers.

Output
Output the list of colors followed by its margin percent to the nearest hundredth (0.00) on a single line. If there is a tie, output the colors in the order they were read in the input.

Input File: C:\DKC3\ProfitIn.txt
Output File: C:\DKC3\ProfitOut.txt

Examples:

Input:
Red 10 20 1
Orange 5 15 2
Yellow 8 24 2
Green 2 24 5
Blue 16 48 4
Purple 1 4 1
*
Red 20 30 1
Orange 5 25 2
Yellow 8 14 2
Green 2 20 3
Blue 7 42 2
Purple 10 14 1
*

Output:
Purple 0.75 Green 0.58 Red 0.50 Orange 0.33 Yellow 0.33 Blue -0.33
Green 0.70 Blue 0.67 Orange 0.60 Red 0.33 Purple 0.29 Yellow -0.14

6. Mean of a Set of Points (10 points)

Finding the smallest circle that encloses a set of points is non-trivial. Therefore, a much simpler method often used by programmers is to find the smallest circle enclosing the points that is centered at the points' mean. Given a set of points as input, your program must find the mean of the set of points to help in this process.

Input
The input will consist of ten test cases, one per line. The test cases will be a set of points in (x, y) format, with a space in between each point.

Output
The output for each test case should be the mean of the set of points, in (x, y) format, where x and y are rounded and expressed to two decimal places.

Input File: C:\DKC3\PointsIn.txt
Output File: C:\DKC3\PointsOut.txt

Examples:

Input:
(0, 0) (4.5, 0) (0, 15)
(-5, 5) (5, -5) (8, 4) (-4, -13)

Output:
(1.50, 5.00)
(1.00, -2.25)

7. Reduction (10 points)

Given a number, determine the smallest possible value created from its digits. You are required to remove one digit from the original number unless the number has only one digit.

Input
There will be ten test cases, one per line. Each test case will contain a number.

Output
For each test case, output a number representing the smallest possible value created from that number (after removing one digit).

Input File: C:\DKC3\ReductionIn.txt
Output File: C:\DKC3\ReductionOut.txt

Examples:

Input:
873
10984

Output:
37
1048

8. Binary Steps (5 points)

Given a binary string ***S***, the task is to print the number of steps required to convert the string down to 1 in binary with the following operations:

- If ***S*** is odd, add 1 to it.
- If ***S*** is even, divide it by 2.

Input
There will be ten test cases, one per line. Each test case will contain a binary string consisting of 1s and 0s.

Output
For each test case, output the number of steps required to convert the binary string to 1 using the operations outlined above.

Input File: C:\DKC3\BinaryStepsIn.txt
Output File: C:\DKC3\BinaryStepsOut.txt

Examples:

Input:
1001001
101110

Output:
12
8

9. Short Hand (15 points)

Write a program that reads a string of fewer than 500 lower-case letters [‘a’…’z’], and finds the character that occurs most often (or, in case of ties, the most frequent character that occurs earliest in the USA Roman alphabet). Remove that character. Print the remaining string. Repeat until the string has no remaining characters, but do not print the empty string.

Input
There will be ten test cases, one per line. Each test case will contain a string of letters no longer than 500 characters.

Output
For each test case, output the string with the most common letter of the previous string removed. Repeat until the string has no remaining characters, but do not print the empty string.

Input File: D:\DKC3\ShortIn.txt
Output File: D:\DKC3\ShortOut.txt

Input/Output examples on the next page.

Examples:

Input:

maryhadalittlelambitsfleecewaswhite
fourscoreandsevenyearsagoourfathersbroughtforth

Output:

mryhdlittlelmbitsfleecewswhite
mryhdlittllmbitsflcwswhit
mryhdittmbitsfcwswhit
mryhdimbisfcwswhi
mryhdmbsfcwswh
mrydmbsfcwsw
rydbsfcwsw
rydbfcww
rydbfc
rydfc
rydf
ryf
ry
y
fouscoeandsevenyeasagooufathesboughtfoth
fusceandsevenyeasagufathesbughtfth
fuscandsvnyasagufathsbughtfth
fuscndsvnysgufthsbughtfth
fucndvnygufthbughtfth
ucndvnyguthbughtth
ucndvnygutbugtt
ucndvnygubug
cndvnygbg
cndvnyb
cdvyb
cdvy
dvy
vy
y

10. Hole in the Bucket (45 points)

Liza asked her husband, Henry, to fill a barrel with water using several different buckets. The barrel must be filled exactly to the top by whole bucket quantities. Henry, being lazy, wants to accomplish the task with as little effort as possible. Write a program to help Henry determine how to fill any size of barrel in the fewest buckets possible as he can only carry one bucket per trip.

Input
Your program input will consist of a list of numbers in a single line. The first number will be the size of the barrel in liters. The following numbers represent the sizes of each available bucket in liters. The number of buckets may vary with each test case. There are ten test cases, one per line.

Output
Output the minimum number of trips using the buckets available to fill the barrel exactly. If there is no exact way to fill the barrel, output 'no solution'.

Input File: C:\DKC3\BucketIn.txt
Output File: C:\DKC3\BucketOut.txt

Examples:

Input:
17 1 2 4 5
21 8 2 4

Output:
4
no solution

11. Metric to US Customary (30 points)

Use the conversion equations below to write a program that will read in a line of text, convert numbers from a metric unit of measure to US customary and print the conversion to the output file.

Helpful conversion equations:

- 1 kilometer = 1,000 meters
- 1 meter = 100 centimeters
- 1 centimeter = 10 millimeters
- 1 inch = 2.54 centimeter
- 1 foot = 12 inches
- 1 yard = 3 feet
- 1 mile = 5,280 feet
- 1 kilogram = 1,000 grams
- 1 ounce = 28.3495 grams
- 1 pound = 16 ounces
- 2 cup = 1 pint
- 2 pint = 1 quart
- 4 quart = 1 gallon
- 1 quart = 0.946353 liters
- 1 liter = 1,000 milliliters

Input/Output information on the next page.

Input
There are ten test cases, one per line. Each test case will contain three values. Value 1 is a decimal number of form x.y where x and y may or may not be zero, and x and y may or may not be the same number. Value 2 is the metric unit of measure of the initial amount. Valid types are: millimeter, centimeter, meter, kilometer, gram, kilogram, milliliter, and liter. Value 3 is the customary unit of measure of the final amount. Valid types are: inch, foot, yard, mile, ounce, pound, cup, pint, quart, and gallon. Types will not be mixed. In other words, the initial type will not be a length (meter), with the final value being a volume (gallon).

Output
For each test case, output the conversion to US Customary. Round and express all answers to five decimal places.

Input File: C:\DKC3\MetricIn.txt
Output File: C:\DKC3\MetricOut.txt

Examples:

Input:
5.5 meter foot
3.0 liter pint

Output:
18.04462
6.34013

12. Earned Run Average (20 points)

In baseball statistics, earned run average (**ERA**) is the average number of earned runs allowed by a pitcher. This can be calculated as the total number of earned runs (**ER**) allowed multiplied by 9 divided by the number of innings pitched (**IP**).

ERA = ((**ER** x 9) / **IP**)

Input
There are ten test cases, one per line. Each test case contains the number of innings pitched, followed by a space, followed by earned runs allowed. The number of innings pitched is displayed as a floating-point number in order to denote partial innings pitched. Since there are 3 outs per inning, it is possible to pitch 1/3, 2/3 or 3/3 innings.

For example:

IP	Description
100.0	100 innings pitched.
100.1	100 + 1/3 innings pitched.
100.2	100 + 2/3 innings pitched.

Output
For each test case, output the calculated ERA on a separate line. Round the results to 2 decimal places. Display the results using 2 decimal places (e.g. 4.10).

Input File: C:\DKC3\ERAIn.txt
Output File: C:\DKC3\ERAOut.txt

Examples:

Input:
116.0 23
96.1 24

Output:
1.78
2.24

13. Seamless (35 points)

Barney is using tongue and groove boards to build some ceilings. He only uses boards which measure 6" front to back with a 1/2" tongue (so they each cover 5.5" of ceiling). He wants to minimize the number of seams, but is only willing to spend so much money.

Help Barney by creating an application he can use to understand how many (minimum) seams a ceiling will have based on the boards he quotes. Also let Barney know if he has enough boards to finish the ceiling.

Note: a 'seam' is where two boards are touching side-to-side without a tongue and groove to keep them together.

Input
There are ten test cases, one per line. Each test case will consist of the following values, each separated by a comma followed by a space: ceiling length in inches, ceiling width in inches, number of 6 ft boards, number of 8 ft boards, number of 12 ft boards.

Output
For each test case, output the number of seams followed by a comma, followed by either "True" or "False" indicating whether Barney has enough boards to finish the ceiling.

Input File: C:\DKC3\SeamlessIn.txt
Output File: C:\DKC3\SeamlessOut.txt

Examples:

Input:
88, 88, 8, 6, 6
88, 88, 1, 1, 1

Output:
4,True
2,False

14. Sequences (40 points)

Write a program that will read in a list of integers and print the next number in the sequence to the output file. Each input line will consist of 4 positive integers separated by a space. Each integer will be no longer than nine digits. There are ten test cases.

Examples:

- An arithmetic sequence is a list of integers with the property that the difference between consecutive integers is constant. Example 2, 4, 6, 8, 10…
- A geometric sequence is a list of integers with the property that the ratio between consecutive integers is constant. Example 2, 4, 8, 16, 32…
- A Lucas sequence is a list of integers with the property that each number, after the second, is the sum of the previous two integers. Example 2, 4, 6, 10, 16, 26…

Input
There are ten test cases, one per line. Each test case will consist of 4 integers separated by a space.

Output
For each test case, output an integer representing the next number in the sequence.

Input File: C:\DKC3\SequenceIn.txt
Output File: C:\DKC3\SequenceOut.txt

Examples:

Input:
2 4 6 8
16 8 4 2

Output:
10
1

DKC^3 2019 – Programming Problems

15. Roman Numerals (25 points)

Write a program that reads in a roman numeral and converts the number to normal decimal. The roman characters are:

M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1

Input
There will be ten test cases, one per line. Each test case will consist of a roman numeral.

Output
For each test case, output a number representing the decimal equivalent of the roman numeral.

Input Files: C:\DKC3\RomanIn.txt
Output Files: C:\DKC3\RomanOut.txt

Examples:

Input:
VIII
XXXIV

Output:
8
34