DKC^3 2017 – Programming Problems

1. Color Switches (20 points)

There is a rectangular grid with N rows and N columns, with (0 < N < 100). Each cell in the grid is one color of the set (Red, White, Blue). Two cells are called adjacent if they share a side. In one move you can choose any two adjacent cells of different colors and change them both to the third color. For example, if two adjacent cells have colors Red and White, then you can change both cells to Blue. You may not choose two non-adjacent cells and you may not choose two cells of the same color. The goal is to switch the cells so that all the cells are of the same color. It doesn't matter with which of the three colors you end.

Input
Each test case will consist of N lines. Each test case will be separated by a blank line. There will be 10 test cases. Each line will have N entries where each entry is one of the three colors separated by a blank space.

Output
1 number that represents the minimum number of switches it will take to convert all the cells in the grid to a single color.

Input File: D:\DKC3\ColorIn.txt
Output File: D:\DKC3\ColorOut.txt

Examples:
Input:
Red White Blue
Blue White Red
White Red Blue

Red Red Red Red
Blue White Red Red
Red Red Red Red
Red Red Red Red

Output:
3
1

2. Parlez Vous? (10 points)

You must take a phrase written in French and translate it into English. You will be given a list of French words and their English equivalent. You will be given a list of French verbs and a short description of how to translate those into English. Then translate the phrases. Easy!?!

Input

There will be 10 lines. Each line is a test case. Each line will be a short phrase in French.

Nouns, adjectives, articles and adverbs:

maison	- house	livre	- book	voiture	- car	avion	- plane
vous	- you	nous	- us	je	- I	Anglais	- English
est	- is	le	- the	la	- the	l'	- the
des	- some	rouge	- red	bleu	- blue	vert	- green
bateau	- boat	au	- to the	sur	- on		

Verb		**Various forms**		**Verb**	**Various forms**	
parler	- speak	je parle	- I speak	peindre – paint	je peins	– I paint
		vous parlez	- you speak		vous peignez	– you paint
		nous parlons	- we speak		nous peignons	– we paint
finir	- finish	je finis	- I finish	aller – to go	je vais	- I go
		vous finissez	- you finish		vous allez	- you go
		nous finissons	- we finish		nous allons	- we go

Output

Print out each French phrase into English

Input File: D:\DKC3\ParlezIn.txt
Output File: D:\DKC3\ParlezOut.txt

Examples:

Input:

Parlez vous Anglais?
L'avion est bleu.

Output:

Speak you English? (The translation should be "Do you speak English?" but let's not get too technical.)
The plane is blue.

3. Repeating Sequences (5 points)

Write a program that will read a four digit integer. From the digits of that integer form the largest possible four digit integer and the smallest possible four digit integer. Calculate the difference of these two integers (largest minus smallest) to form a new four digit integer. Then repeat the process until the new four digit integer created by the subtraction is the same as the four digit integer you just used.

Input
There will be 10 test cases. Each test case will be a single line containing an integer of four digits. None of the integers will start with a 0, however there will be 0's within some of the numbers.

Output
Each output line should include the initial integer followed by the rest of the sequence. Each integer should be separated by a space. Each integer in the sequence should be formatted as a four digit integer. If the subtraction results in a 3 digit number you must assume that number starts with a zero. For example: 2111 – 1112 = 999 but you need a four digit integer for the next subtraction so the 999 becomes 0999. Only output the repeating integer once as the last integer of the sequence.

Input File: D:\DKC3\RepeatIn.txt
Output File: D:\DKC3\RepeatOut.txt

Examples:
Input:
8765
8333

Output:
8765 3087 8352 6174
8333 4995 5355 1998 8082 8532 6174

4. Phoneme Palindromes (10 points)

A palindrome is a string that reads the same forward and backward, e.g., madam and abba. Since some letters sound the same (e.g., c and k), we define a phoneme palindrome as a string that sounds the same forward and backward, e.g., cak and ckckbbkcck. Given the letters that sound the same and a string, you are to determine if the string is a phoneme palindrome.

Input
There will be 10 test cases. Each test case starts with an integer, p (1 <= p <= 13) , indicating the count for pairs of letters that sound the same. Each of the following p input lines provides two distinct lowercase letters (starting in column 1 and separated by a space) that sound the same. Assume that no letter appears in more than one pair. The next input line for a test case contains an integer, q (1 <= q <= 100), indicating the number of strings to test for phoneme palindrome. Each of the following q input lines provides a string (starting in column 1 and lowercase letters only) of length 1 to 50, inclusive.

Output
For each test case, print the header "Test case #n:", where n indicates the case number starting with 1. Then print each string for that test case followed by a space, followed by a message (YES or NO) indicating whether or not the string is a phoneme palindrome. Leave a blank line after the output for each test case.

Input File: D:\DKC3\PalIn.txt
Output File: D:\DKC3\PalOut.txt

Examples:

Input:

1
c k
6
a
cac
ck
cab
kaak
ckckkcck
2
a z
x s
5
abbbz
asxz
cx
sxxabzxss
ks

Output:

Test case #1:
a YES
cac YES
ck YES
cab NO
kaak YES
ckckkcck YES

Test case #2:
abbbz YES
asxz YES
cx NO
sxxabzxss YES
ks NO

5. NIH Budget (20 points)

Recently, a job for an algorithms specialist opened at National Institutes of Health (NIH). You never thought you'd be using your expertise in algorithms to save lives, but now, here is your chance! While the doctors are very good in carrying out medical research and coming up with better cures for diseases, they are not so good with numbers. This is where you come in.

You have been tasked to allocate money for all disease research at NIH. The interesting thing about disease research is that the number of lives saved doesn't linearly increase with the amount of money spent, in most cases. Instead, there are "break-points". For example, it might be the case that for disease A, we have the following break-points:

Research Funding	Lives Saved
10 million	5
50 million	100
100 million	1000
250 million	1100

If you spend more money than one breakpoint and less than another, the number of lives saved is equal to the amount saved for the previous breakpoint. (In the above example, if you spent $150 million, you'd still only save 1000 lives, and if you spent any amount more than $250 million, you'd still save 1100 lives.)

The doctors have figured out charts just like this one for all the diseases for which they do research. Given these charts, your job will be to maximize the number of lives saved spending no more than a particular budget.

Given several charts with information about how much must be spent to save a certain number of lives for several diseases and a maximum amount of money you can spend, determine the maximum number of lives that can be saved.

Input
There are ten test cases (i.e. budgets). The first line of each budget contains two positive integers: d (d ≤ 10), representing the number of diseases for which there is data, and B (B ≤ 100000), the total budget in millions of dollars. The following d lines contain information about each of the d diseases. Each of these lines will contain ordered pairs of positive integers separated by spaces. Each line will end with a newline character. Each pair will represent a dollar level (in millions) followed by the number of lives saved for that dollar level of funding. Each of the pairs will be separated by spaces as well. Each of these values will be less than or equal to 100,000. Assume that the dollar levels on an input line are distinct and in increasing order, and that the number of lives saved on an input line are also distinct and in increasing order.

Output
For each test case, just output a line with the following format:

Budget #k: Maximum of x lives saved.

where k is the number of the budget, starting at 1, and x is the maximum number of lives saved in that budget.

Input File: D:\DKC3\BudgetIn.txt
Output File: D:\DKC3\BudgetOut.txt

Examples:

Input:
2 2000
10 5 50 100 100 1000 250 1100
100 1 200 2 300 3 1900 1000
1 10
100 2 200 3 300 5 400 6

Output:
Budget #1: Maximum of 2000 lives saved.
Budget #2: Maximum of 0 lives saved.

6. Sudoku Summation (25 points)

Sudoku is the name given to a popular puzzle concept. The rules of the game are simple: each of the nine blocks must contain all the numbers 1-9 within its squares. Each number can only appear once in a row, column or box. Below is an example of a typical starting puzzle grid and its solution grid.

0 0 3	0 2 0	6 0 0
9 0 0	3 0 5	0 0 1
0 0 1	8 0 6	4 0 0
0 0 8	1 0 2	9 0 0
7 0 0	0 0 0	0 0 8
0 0 6	7 0 8	2 0 0
0 0 2	6 0 9	5 0 0
8 0 0	2 0 3	0 0 9
0 0 5	0 1 0	3 0 0

4 8 3	9 2 1	6 5 7
9 6 7	3 4 5	8 2 1
2 5 1	8 7 6	4 9 3
5 4 8	1 3 2	9 7 6
7 2 9	5 6 4	1 3 8
1 3 6	7 9 8	2 4 5
3 7 2	6 8 9	5 1 4
8 1 4	2 5 3	7 6 9
6 9 5	4 1 7	3 8 2

A well-constructed Sudoku puzzle has a unique solution and can be solved by logic, although it may be necessary to employ "guess and test" methods in order to eliminate options. The complexity of the search determines the difficulty of the puzzle; the example above is considered easy because it can be solved by straight forward direct deduction.

Input
For each test case there will be 1 to 10 Sudoku puzzles that must be solved. By solving each of the puzzles in a test case, find the sum of the 3-digit numbers found in the top left corner of each solution grid; for example, 483 is the 3-digit number found in the top left corner of the solution grid above, and since there is only one, the output would be 483.

The input will come in the form of a 9x9 matrix for each puzzle. Each puzzle within a group shall be separated by a blank line, and each group shall be separated by an asterisk (*).

Output
Output should be sent to the file, one answer per line.

Input File: D:\DKC3\SudokuIn.txt
Output File: D:\DKC3\SudokuOut.txt

Examples:

Input:

003020600
900305001
001806400
008102900
700000008
006708200
002609500
800203009
005010300
*
200080300
060070084
030500209
000105408
000000000
402706000
301007040
720040060
004010003

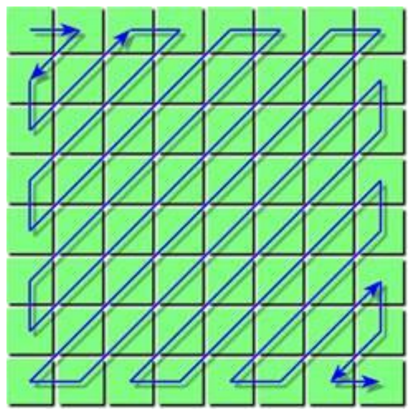
000000907
000420180
000705026
100904000
050000040
000507009
920108000
034059000
507000000
*

Output:

483
707

7. Ziggity Zag (5 points)

A zig-zag array is a square arrangement of the first N2 integers, where the numbers increase sequentially as you zig-zag along the array's anti-diagonals. For a graphical representation, refer to the following image.



For example, given n = 5, the following array is produced.

0	1	5	6	14
2	4	7	13	15
3	8	12	16	21
9	11	17	20	22
10	18	19	23	24

Input
There will be 10 test cases for this problem, one test case per line. You will be given a number, n, which will represent the size of the matrix to be filled with incrementing integers in a zig-zag pattern starting from 0 (zero). There will be one value per line in the input file where 2 < n < 100000.

Output
With the provided information, find the sum of the perimeter values and output the final value to the output file, one value per line. With the above grid, the sum would be 192.

Input File: D:\DKC3\ZiggityIn.txt
Output File: D:\DKC3\ZiggityOut.txt

Examples:

Input:

5
10

Output:

192
1782

DKC^3 2017 – Programming Problems

8. Order of Operations Parser (10 points)

In mathematics and computer programming, the order of operations (or operator precedence) is a collection of rules that reflect conventions about which procedures to perform first in order to evaluate a given mathematical expression.

While computers tend to help us with these operations automatically, it is sometimes necessary to evaluate these expressions by hand, which is where PEMDAS can be useful.

PEMDAS is an acronym for the words parenthesis, exponents, multiplication, division, addition, subtraction. Given two or more operations in a single expression, the order of the letters in PEMDAS tells you what to calculate first, second, third and so on, until the calculation is complete.

For this problem, you will be given an expression that must be parsed in the proper order and the solution found.

An example would be the following string.

(2 + 3) * 3 – 5 + 10 / 2 = X

You must write a program to solve for X.

5 * 3 – 5 + 5
15 – 5 + 5 = 15
X = 15

Input
The input for this problem will consist of 10 test cases, one per line. Each line will have one equation which must be parsed and the final answer output to the file in the form of the character being solved for, followed by an equals sign "=" with a space on either side, and then the value. Such as "X = 15", or "Y = 23", and so forth. The character will be placed in the equation and must be located. Note that there may be operators on both sides of the equals sign, but you will only be solving for one character at a time.

Output

Input File: D:\DKC3\OrderOpIn.txt
Output File: D:\DKC3\OrderOpOut.txt

Examples:

Input:

(2 + 3) * 3 – 5 + 10 / 2 = X
b + 5 = 20 – (3 + 7) / 2 - 10

Output:

X = 15
b = 0

9. Mustache Melon (5 points)

Limey Simon sells 5 very popular mustache waxes. The prices for these items are:

MAXWAX22 = $4.10

LEMONSCENT-77= $9.60

MOUNTAINEER-MN-18 = $6.78

BROOKLYNBURG-19 = $12.25

CLUBMAN-GA-15 = $8.60

Write a program that reads a list of waxes and quantities sold for one day. Your program should calculate and display the total sales for the items for that day, where quantity is greater than zero and less than 65536. You can assume that if a wax is listed, the quantity for the item will be greater than zero. In other words, if an item wasn't sold that day, it won't be listed in the input.

Input
Each test case will consist of several lines of input containing an item number followed by a single space and then a quantity. There will be 10 test cases, each separated by an asterisk.

Output
For each test case, you should output a dollar amount, including the dollar sign and decimal and comma separated 1000's. The answers must contain 2 decimal places.

Input File: C:\DKC3\WaxIn.txt
Output File: C:\DKC3\WaxOut.txt

Examples:

Input:

```
MAXWAX22 98
BROOKLYNBURG-19 201
CLUBMAN-GA-15 1
*
LEMONSCENT-77 2
*
```

Output:

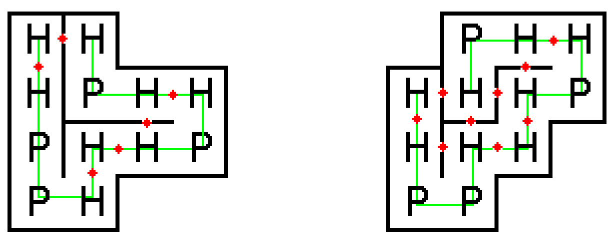
```
$2,872.65
$19.20
```

10. Dare to Fold (35 points)

In a very simplified form, we can consider proteins as strings consisting of hydrophobic (H) and polar (P) elements, e.g. HHPPHHHPHHPH.
For this problem, the orientation of a protein is important; e.g. HPP is considered distinct from PPH.
Thus, there are 2n distinct proteins consisting of n elements.

When one encounters these strings in nature, they are always folded in such a way that the number of H-H contact points is as large as possible, since this is energetically advantageous.
As a result, the H-elements tend to accumulate in the inner part, with the P-elements on the outside.
Natural proteins are folded in three dimensions of course, but we will only consider protein folding in two dimensions.

The figure below shows two possible ways that our example protein could be folded (H-H contact points are shown with red dots).



The folding on the left has only six H-H contact points, thus it would never occur naturally.
On the other hand, the folding on the right has nine H-H contact points, which is optimal for this string.

Assuming that H and P elements are equally likely to occur in any position along the string, the average number of H-H contact points in an optimal folding of a random protein string of length 8 turns out to be $850 / 2^8 = 3.3203125$.

Input
Givin a certain length what is the average number of H-H contact points in an optimal folding of a random protein string

Output
Give your answer using as many decimal places as necessary for an exact result

Input File: D:\DKC3\FoldIn .txt
Output File: D:\DKC3\FoldOut .txt

Examples:

Input:

5

8

Output:

1.4375

3.3203125

11. ECIA Compliant Barcodes (20 points)

When a manufacturer barcode is scanned at Digi-Key, there are several pieces of information that can be retrieved from the barcode. One of the first things we check for is to verify that a barcode is ECIA compliant (Electronic Components Industry Association). In order for a barcode to be ECIA compliant, it must have a specific header/trailer, have proper identifiers, and have all pieces of data separated with the correct delimiters. Below are the different Data Identifiers and Ascii Delimiters used when creating ECIA compliant barcodes.

Data Identifier	Field Name
P	Customer Part Number
1P	Manufacturer Part Number
9D, 10D, 11D	Date Code
Q	Quantity
1T	Lot Code
4L	Country of Origin

Ascii Char #	Name	Text Representation
4	End Of Transmission	**[EOT]**
29	Group Separator	**[GS]**
30	Record Separator	**[RS]**

The header for an ECIA compliant barcode will be in the format [)>[RS]06[GS] where [RS] and [GS] are the record separator character (char 30) and the group separator (char 29) respectively. The header will immediately be followed by a data identifier (some of which are listed above), then followed by the contents of that identifier terminating at the next group separator [GS] (i.e. if you had [GS]1P1234-PART[GS], your data identifier would be 1P, corresponding to the manufacturer part number, and the content would be 1234-PART). Finally, the trailer will indicate the end of the given barcode with a record separator character, [RS] (char 29) followed immediately by an end of transmission character, [EOT] (char 4).

Input
Following the information above, write a program that reads in a barcode (1 barcode per line), parses out the Customer Part Number, the Manufacturer Part Number, Date Code, Quantity, Lot Code, and Country of Origin, and outputs as a comma separated list, with no spaces between each value, as shown in the examples below.

Output
The output data should be displayed in the same order as they are listed on the Data Identifier table above. Also, there can be more data identifiers/fields than the ones listed above, these should not be listed in the output. There will be one test case per line. There will be ten test cases.

Input File: C:\DKC3\ECIACompliantIn.txt
Output File: C:\DKC3\ECIACompliantOut.txt

Examples:

Input:

[)>**[RS]**06**[GS]**PPART_NUM123**[GS]**1PMAN_PART_NUM321**[GS]**9DDATE_CODE**[GS]**Q1**[GS]**1TLOT_CODE**[GS]**4LUS**[GS][RS][EOT]**
[)>**[RS]**06**[GS]**PERJ_8ENF1003V**[GS]**1PERJ8ENF1003V**[GS]**Q5000**[GS]**10D170520**[GS]**1T9452268475**[GS]**4LJP**[GS][RS][EOT]**

Output:

PART_NUM123,MAN_PART_NUM321,DATE_CODE,1,LOT_CODE,US
ERJ_8ENF1003V,ERJ8ENF1003V,170520,5000,9452268475,JP

12. Counting Sundays (5 points)

Given the following information.

- 1 Jan 1900 was a Monday.
- Thirty days has September,
  April, June and November.
  All the rest have thirty-one,
  Saving February alone,
  Which has twenty-eight, rain or shine.
  And on leap years, twenty-nine.
- A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

Count how many times a certain date of the month fell on a given day.

Input
You will first be given the day of the week you are to be searching for (following the below chart). You will then be given the date of the month we wish to find that day on. For example, how many Sundays fell on the first of the month? Each test case will be on its own line with space separated values.

Sunday – 1
Monday – 2
Tuesday – 3
Wednesday – 4
Thursday – 5
Friday – 6
Saturday – 7

Output
You will need to output the number of those days found on that day of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000). Each answer must be on its own line.

Input Files: D:\DKC3\CountingIn.txt
Output Files: D:\DKC3\CountingOut.txt

Examples:

Input:
6 13
2 31

Output:
171
99

13. Fibonacci Sequence (15 points)

The Fibonacci sequence is defined by the recurrence relation:
$F_n = F_{n-1} + F_{n-2}$, where $F_1 = 1$ and $F_2 = 1$.

Hence the first 12 terms will be:
$F_1 = 1$
$F_2 = 1$
$F_3 = 2$
$F_4 = 3$
$F_5 = 5$
$F_6 = 8$
$F_7 = 13$
$F_8 = 21$
$F_9 = 34$
$F_{10} = 55$
$F_{11} = 89$
$F_{12} = 144$

The 12th term, F_{12}, is the first term to contain three digits.

Input:
Each line will have a separate test case containing a single digit. This single digit corresponds to how many digits long the number in the Fibonacci sequence needs to be.

Output:
You will need to provide the term that first appears in the sequence with that number of digits. There will be ten test cases. Using the above as an example, the input would be 3. The 12th term is the first to contain 3 digits. Therefore, 12 is the output.

Input Files: D:\DKC3\FibonacciIn.txt
Output Files: D:\DKC3\FibonacciOut.txt

Examples:

Input:
10
6
18

Output:
45
26
84

14. Permuted Multiples (10 points)

A permuted multiple can be described as a set of numbers *[n, m]* where *m* is a multiple of *n* and both numbers consist of the same digits in a different order. For example, the number 125874 (in this case, *n*) and its double 251748 (*m*) contain exactly the same digits, but in a different order. So [125874, 251748] is an acceptable 2x permuted multiple due to *m* being twice the *n*.

Input
Each test case will be on its own line and consist of two numbers .The first number, x, will represent the multiple difference between *m* and *n*. The second number will represent p that satisfies the statement of "the p lowest permutated set [n, m]".

Output
For the purposes of this question, we're only asking for positive, non-decimal integers.
Find the p lowest set [n, m] where n is x the m. Output expected in the format [n, m].

Input Files: D:\DKC3\MultiplesIn.txt
Output Files: D:\DKC3\MultiplesIn.txt

Examples:
Input:
4 2
6 3

Output:
[2178, 8712]
[13986, 83916]

15. Divisibility Streaks (5 points)

For every positive number *n* we define the function *streak(n)=k* as the smallest positive integer *k* such that *n+k* is not divisible by *k+1*.

E.g.:
13 is divisible by 1
14 is divisible by 2
15 is divisible by 3
16 is divisible by 4
17 is NOT divisible by 5

So streak(13)=4.

Input
You will be given the value *n* in a single line test case.

Output
Output the subsequent *k* value in the format *streak(n)=k*. Such that *n* is the input value given and *k* is the streak value calculated.

Input Files: D:\DKC3\DivisibilityIn.txt
Output Files: D:\DKC3\DivisibilityOut.txt

Examples:

Input:
7
1115

Output:
streak(7)=2
streak(1115)=4