DKC^3 2016 – Short Computing Problems

1. Reverse Factorial (5 points)

The factorial of an integer n (written as n!) is the product of all the integers from 1 to n inclusive. For example, 4! = 1 * 2 * 3 * 4 = 24.

Write a program that accepts a number between 1 and 9223372036854775807 inclusively and determines whether or not that number is a product of a factorial. If the number is a product of a factorial, output the original factorial. If the number is not a product of a factorial, output the word 'none'. There will be ten test cases, one test case per line.

Input File: D:\DKC3\FactorialIn.txt
Output File: D:\DKC3\FactorialOut.txt

Examples:

Input:

```
24
362880
5000
```

Output:

```
4
9
none
```

2. Reverse Polish Notation (10 points)

Reverse polish notation is the notation where the operators follow the operands (AKA Postfix). For example “3 + 4” is written “3 4 +” in Reverse Polish Notation.

Write a program that reads in equations represented in Reverse Polish Notation and then outputs the solution to the equation. A space will separate each number/operator. Only the binary operators +, -, *, / will be used. There will be ten test cases, one test case per line.

Input File: D:\DKC3\ReversePolishIn.txt
Output File: D:\DKC3\ReversePolishOut.txt

Examples:

Input:

```
3 4 * 2 +       “this represents the equation: 3*4+2”
2 3 4 + -       “this represents the equation: 2-(3+4)”
```

Output:

```
14
-5
```

3. The Biggest (35 points)

The Mighty Modulas have been at it again: They've been given index cards printed with numbers on both side, and they've torn each card between the digits. Your job, as virtual leader of the Mighty Modulas, is to teach your charges how to tear each card so that the numbers on all the scraps add up as close as possible to some given target number without going over. You are free, of course, to turn over any scrap.

Consider, for example, a card with the number 534 on one side and 197 on the other side. There are 3 ways that you could tear the card: between the 5 and the 3; between the 3 and the 4; and after the 5 and after the 3. In the first case, you have two strips: one with 5 on one side and 7 on the other, and the other strip with 34 on one side and 19 on the other. In the second case, you'd also have two strips: one with 53 on one side and 97 on the other, and the second string with 4 on one side and 1 on the other. Finally, in the third case, you'd have 3 strips. You must make at least one tear. If the numbers are different lengths, pad the shorter one with leading zeros so that they are of the same length.

You'll be given 10 sets of data. Each set consists of three number: the number on one side of the card, the number on the other side of the card, and a target number. You are to figure out how to tear the card such that the number one side or the other of the torn scraps add up as close as possible to the target number without going over. Output the sum. The number on the cards will be positive integers less than 1,000,000,000.

Input File: D:\DKC3\BigIn.txt
Output File: D:\DKC3\BigOut.txt

Examples:

Input:

```
534, 197, 50
534, 197, 100
```

Output:

```
41
98
```

4. Ana's Anagram (15 points)

Ana spends her time looking for patterns. She finds amusement in counting anagrams found in newspaper articles, books, and poetry at a local coffee shop.

Anagrams are a word or phrase that is formed by rearranging the letters of another word or phrase. The rules Ana uses to determine whether or not an anagram exists within paragraphs are fairly simple and are as follows:

- Begin at the start of a paragraph and end on the last sentence of a paragraph.
- Omit punctuation in the anagram evaluation.
- A complete sentence must be an anagram of another complete sentence within the same paragraph to count as one of Ana's Anagrams, not word to word.
- Tally up the number of unique anagram sentences found in each paragraph. For example, if three complete sentences are anagrams of each other it would be considered one unique anagram.

You will be given ten paragraphs as input. Each paragraph is separated by a blank line. Output the number of Ana's Anagrams found per each paragraph on its own line.

Input File: D:\DKC3\AnagramIn.txt
Output File: D:\DKC3\AnagramOut.txt

Examples:

Input:

```
Eleven plus two.
Twelve plus one.

A rolling stone gathers no moss.
Stroller on go, amasses nothing.
Silicon Graphics.
A long chip crisis.
Can logic ship, sir?
```

Output:

```
1
2
```

5. Spiral (15 points)

Starting with the number 1 and moving to the right in a clockwise direction a 5 by 5 spiral is formed as follows:

```
21 22 23 24 25
20  7  8  9 10
19  6  1  2 11
18  5  4  3 12
17 16 15 14 13
```

Write a program where given a single input (x) with a range 1 -> 999999 inclusive, output the sum of the numbers on the diagonals in a (x) by (x) spiral formed in the same way as the above example. There will be 10 test cases, one test case per line.

Input File: D:\DKC3\SpiralIn.txt
Output File: D:\DKC3\SpiralOut.txt

Examples:

Input:

5
1001

Output:

101
669171001

6. Binary Bridge (10 points)

A binary bridge is a sequence of consecutive ones within a binary number where there is at least one zero before the start of the sequence and at least one zero at the end of the sequence.

For instance, the number 6 has binary representation 0110 and contains a binary bridge of length 2. The number 16285 has the binary representation 0011 1111 1001 1101 and contains two binary bridges: one of length 7 and one of length 3. The number 48 has the binary representation 0011 0000 and contains one binary bridge of length 2 (be cognizant of four bit sequence).

Write a program that reads in a positive integer and outputs the longest binary bridge for that number. There will be ten test cases, one test case per line.

Input File: D:\DKC3\BridgeIn.txt
Output File: D:\DKC3\BridgeOut.txt

Examples:

Input:

```
16285
48
```

Output:

```
7
2
```

7. Pyramid Scheme (30 points)

A pyramid of numbers where row size 3 < N < 75 must be traversed by starting at the top row and moving to an adjacent number on the row below it. As you traverse the pyramid sum the numbers taking the path that provides the maximum sum.

```
   **3**
  **7** 4
 2 **4** 6
8 5 **9** 3
```

That is, 3 + 7 + 4 + 9 = 23.

For example, the path through the above pyramid would be 3 + 7 + 4 + 9 = 23 rather than 3 + 4 + 4 + 3 = 14.

Write a program that reads in a pyramid, determines the path that provides the maximum sum and then outputs those numbers followed by the sum of the numbers. There will be ten test cases and each test case will be separated by a blank line.

Input File: D:\DKC3\PyramidIn.txt
Output File: D:\DKC3\PyramidOut.txt

Examples:

Input:

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

30
10 43
1 27 93
17 88 88 27
55 67 28 48 93
50 35 4 89 20 50
```

Output:

```
3 7 4 9 23
30 43 93 88 48 89 391
```

8. Most Valuable Word (5 points)

Write a program that will read in a string of words, find the word that when each character is converted to its ASCII value, and each character in the word is added up, results in the highest value in that line. The words will all be in lower case, so the ASCII values will be between 97 and 122 inclusive.

Each test case will contain between 1 and 10 words, and the words will be no longer than 50 characters each. The words will be separated by spaces, but no other whitespace or punctuation characters will be used. There will be ten test cases, one test case per line.

For each test case, output the word that has the highest total ASCII value. Separate each line of output with a newline character.

Input File: D:\DKC3\WordIn.txt
Output File: D:\DKC3WordOut.txt

Examples:

Input:

```
the quick brown fox jumps over the lazy dog
lorem ipsum dolor sit amet
```

Output:

```
jumps
ipsum
```

9. Bowling... But Different (20 points)

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.

Let's say the number of frames and the number of pins per frame were able to be changed. Write a program that, following the rules for each frame above (where the only exceptional case is the final frame), accepts an input of a list of integers separated by commas and has an output of 1 integer. The first input integer is the number of frames, the second input integer is the number of pins per frame, and every integer following that is the bowler's rolls. Keep in mind that if a bowler gets a strike (knocks all the pins down on the first roll), there will only be 1 roll for that frame. For each test case output he final score that the bowler rolled. There will be ten test cases, one test case per line.

Input File: D:\DKC3\BowlingIn.txt
Output File: D:\DKC3\BowlingOut.txt

Examples:

Input:

```
10,10,6,4,10,5,5,8,2,8,1,10,10,10,10,8,2,8
7,15,15,14,1,10,2,6,9,15,9,4,15,15,15
```

Output:

```
211
183
```

10. Roman Checkers (15 points)

Roman numerals represent a numeric system originating from ancient Rome. Numbers in this system are represented by combinations of letters from the Latin alphabet and, as they're used today, are based on seven symbols:

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1,000

To write a number in Roman numerals, the numerals are written left to right in descending order of value and, to find the number represented by Roman numerals, the values of the numerals are summed, such as MMXVI (two thousands, one ten, one five, and one one) is 2016.

To avoid four characters repeated in succession such as IIII, XXXX, and CCCC, the numeral that would be repeated is positioned to the left of the next numeral in value such as IV, XL, and CD, respectively. The ones, tens, hundreds, and thousands are treated as separate items so 99 is written as XCIX, 90 + 9, but it should never be written as IC. Likewise, 999 is written as CMXCIX, not IM, and 45 is written as XLV, not VL.

However, some Romans didn't adhere to these strict conventions of writing their numbers, and they used improper additive, such as IIII for 4, and subtractive, such as LM for 950, forms, sometimes even using double subtractive, such as IIX for 8. These numbers were still readable but didn't adhere to the proper form, and they were appropriately named improper Roman numerals.

Write a program that will take in a number written in Roman numerals. If the number is written in improper Roman numerals, output the proper Roman numerals to represent that number. If the number is already in the proper form, output the value of that number. There will be ten test cases, one test case per line.

Input File: D:\DKC3\RomanIn.txt
Output File: D:\DKC3\RomanOut.txt

Examples:

Input:

```
XV
IM
```

Output:

```
40
CMXCIX
```

11. Botchagaloop (10 points)

The botchagaloop value of a number x is found as follows. First, convert x to base 8. Call this p. Next, sort the digits of p in increasing order. Call this q. Subtract q from p (in base 8, of course). Repeat the "sort-subtract" sequence 4 more times, or until the digits in the result are in sorted order (whichever comes first). Finally, covert the number back to base 10.

For example 3418 has a botchagaloop value of 1008. It is computed as follows: 3418 = 6532; 6532 - 2356 = 4154; 4154 - 1445 = 2507; 2507 – 257 = 2230; 2230 - 223 = 2005; 2005 – 25 = 1760; and finally, 1760 = 1008. Note that there is at least one subtraction and at most 5 subtractions.

Write a program that reads in a number and outputs the botchagalopp value. Each test case is a positive integer less than 1,000,000. There will be ten test cases, one test case per line.

Input File: D:\DKC3\BotchagaloopIn.txt
Output File: D:\DKC3\BotchagaloopOut.txt

Examples:

Input:

```
3418
123
```

Output:

```
1008
28
```

12. Folland Compression (5 points)

This problem involves writing a program to compress or decompress a list of 10 integers by implementing the Folland Compression algorithm. The input to this program will be the word COMPRESS or DECOMPRESS followed by a list of 10 integers, where each integer is between -1,000,000 and 1,000,000. You are to implement the following compression scheme: To compress the list, leave the first integer as it is and in place of the second integer, output the difference between the first and second integers. In place of the third integer, output the difference between the original second and third integers, etc. The decompression algorithm is simply the reverse operation of the compression process, where the list of integers is restored to its uncompressed format. There will be ten test cases, one test case per line.

Input File: D:\DKC3\ FollandIn.txt
Output File: D:\DKC3\ FollandOut.txt

Examples:

Input:

```
COMPRESS 1768 1779 1771 1780 1786 1790 1795 1801 1799 1803
DECOMPRESS 1768 11 -8 9 6 4 5 6 -2 4
```

Output:

```
1768 11 -8 9 6 4 5 6 -2 4
1768 1779 1771 1780 1786 1790 1795 1801 1799 1803
```

DKC^3 2016 – Short Computing Problems

13. Connect Four (20 points)

In the 20×20 grid below, four numbers along a diagonal line have been marked in red. The product of these numbers is 26 × 63 × 78 × 14 = 1788696.

08	02	22	97	38	15	00	40	00	75	04	05	07	78	52	12	50	77	91	08
49	49	99	40	17	81	18	57	60	87	17	40	98	43	69	48	04	56	62	00
81	49	31	73	55	79	14	29	93	71	40	67	53	88	30	03	49	13	36	65
52	70	95	23	04	60	11	42	69	24	68	56	01	32	56	71	37	02	36	91
22	31	16	71	51	67	63	89	41	92	36	54	22	40	40	28	66	33	13	80
24	47	32	60	99	03	45	02	44	75	33	53	78	36	84	20	35	17	12	50
32	98	81	28	64	23	67	10	**26**	38	40	67	59	54	70	66	18	38	64	70
67	26	20	68	02	62	12	20	95	**63**	94	39	63	08	40	91	66	49	94	21
24	55	58	05	66	73	99	26	97	17	**78**	78	96	83	14	88	34	89	63	72
21	36	23	09	75	00	76	44	20	45	35	**14**	00	61	33	97	34	31	33	95
78	17	53	28	22	75	31	67	15	94	03	80	04	62	16	14	09	53	56	92
16	39	05	42	96	35	31	47	55	58	88	24	00	17	54	24	36	29	85	57
86	56	00	48	35	71	89	07	05	44	44	37	44	60	21	58	51	54	17	58
19	80	81	68	05	94	47	69	28	73	92	13	86	52	17	77	04	89	55	40
04	52	08	83	97	35	99	16	07	97	57	32	16	26	26	79	33	27	98	66
88	36	68	87	57	62	20	72	03	46	33	67	46	55	12	32	63	93	53	69
04	42	16	73	38	25	39	11	24	94	72	18	08	46	29	32	40	62	76	36
20	69	36	41	72	30	23	88	34	62	99	69	82	67	59	85	74	04	36	16
20	73	35	29	78	31	90	01	74	31	49	71	48	86	81	16	23	57	05	54
01	70	54	71	83	51	54	69	16	92	33	48	61	43	52	01	89	19	67	48

There will be 10 test cases, each test case consisting of a 20x20 grid. For each test case find the greatest product in the grid that consists of 4 numbers that are adjacent to each other either vertically, horizontally, or diagonally.

The output format will start with the direction of the four adjacent numbers; Vertical, Horizontal or Diagonal and immediately followed by a colon. Then, the four numbers that produce the highest product will be displayed along with the answer in standard arithmetic form with spacing. One answer per line.

{DIRECTION}: {NUM1} * {NUM2} * {NUM3} * {NUM4} = {ANSWER}

Input File: D:\DKC3\ConnectFourIn.txt
Output File: D:\DKC3\ConnectFourOut.txt

Examples:

Input:

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

63 72 16 41 17 57 19 45 88 39 38 76 94 00 98 61 19 10 62 89
93 67 73 50 39 30 09 64 97 87 81 93 46 22 95 67 78 47 08 61
90 74 62 50 75 93 38 53 38 62 76 17 48 09 38 39 02 63 69 54
29 69 59 15 73 67 11 82 65 89 06 61 50 60 79 08 22 41 85 67
06 59 85 59 95 28 54 75 83 13 58 11 71 82 38 83 62 63 14 80
46 46 33 46 40 16 95 05 66 10 88 19 60 26 69 35 53 47 12 51
62 94 51 20 90 74 34 80 79 58 42 71 30 61 08 83 00 45 18 87
64 95 21 53 93 22 22 13 83 42 35 00 17 30 21 77 01 55 97 36
86 74 26 32 94 82 67 50 75 20 75 05 70 74 88 82 33 92 50 72
92 06 46 03 99 70 27 60 14 93 64 88 43 06 26 32 82 54 23 51
06 64 52 71 64 72 39 99 83 53 58 65 40 63 42 49 83 59 83 80
30 85 89 84 88 37 94 67 92 89 82 36 89 56 56 59 92 43 09 15
83 30 09 22 13 65 86 13 06 56 50 17 93 57 60 81 59 53 63 99
71 51 09 91 95 94 68 12 00 88 84 85 03 80 86 34 47 04 22 38
68 21 15 85 33 37 16 16 52 80 18 18 73 48 53 26 40 34 45 81
50 64 17 93 02 54 35 40 50 93 80 49 98 19 60 27 19 29 34 16
61 71 28 24 92 50 19 05 42 28 37 49 87 87 18 72 09 29 04 43
70 32 56 02 03 75 71 86 62 96 57 76 40 80 08 13 62 50 91 22
46 91 79 16 09 36 26 42 02 83 04 45 90 08 05 52 71 74 42 46
40 88 63 15 23 94 06 89 03 47 03 14 71 60 99 61 41 35 24 55
```

Output:

```
Diagonal: 87 * 97 * 94 * 89 = 70600674
Vertical: 90 * 93 * 94 * 99 = 77891220
```

DKC^3 2016 – Short Computing Problems

14. Base Numbers (15 points)

It is possible to convert integers between any base number system. For example, 5 in decimal (base 10) is equivalent to 101 in binary (base 2). For bases that have more than 10 unique digits, uppercase letters of the English alphabet can be used, starting with 'A' and working towards 'Z' for the additional digits. This allows us to represent numbers from base 2 up to base 36.

Write a program that converts a series of numbers from a given base to another specified base.

The input consists of one line of text per test case. The first set of characters will be a string that represents a number in a given base. There will then be a tab ('\t') character, and a number that specifies which base the preceding number is currently in. Then there will be another tab that specifies the base that the number needs to be converted to. Note, both of the specifiers will themselves be in base 10. Also, you can assume that if the number were to be converted to base 10, it would be able to fit in an unsigned 32-bit integer.

For each test case, output the given number in the specified base number system. There will be 10 test cases, one test case per line.

Input File: D:\DKC3\ BaseIn.txt
Output File: D:\DKC3\BaseOut.txt

Examples:

Input:

```
1010	2	10
1022	3	16
```

Output:

```
10
23
```

DKC^3 2016 – Short Computing Problems

15. Friday the 13th (20 points)

Steve is a superstitious prepper that believes zombies will be unleashed upon the earth on Friday the 13th. His underground habitat is in a constant state of preparation where he will wait out the apocalypse. He knows there is a chance that the apocalypse will not happen and doesn't want to lose his job so he takes the day off from work just in case.

Write a program that reads in a start year and end year and then outputs the number of days Steve needs to take off from work starting from Jan 1st of the start year and ending on Dec 31st of the end year. There will be ten test cases, one test case per line.

For example given Input of:

2010 2012

represents a start date of Jan 1st 2010 and end date of Dec 31st 2012. There are two Friday the 13ths between those dates falling in August of 2010 and May of 2011. The output for this example is 2.

Input File: D:\DKC3\FridayIn.txt
Output File: D:\DKC3\FridayOut.txt

Examples:

Input:

```
2000    2100
2016    2058
```

Output:

```
162
95
```