Computer Science Competition Invitational B 2019 Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Alfonso
Problem 2	Barb
Problem 3	Bryce
Problem 4	Eduardo
Problem 5	Goro
Problem 6	Harmony
Problem 7	Hyun
Problem 8	Joaquin
Problem 9	Mary
Problem 10	Nisha
Problem 11	Riley
Problem 12	Sara

1. Alfonso

Program Name: Alfonso.java **Input File: None**

Alfonso was playing with for loops and strings and came up with the following pattern. Can you recreate it with for loops? Do you need to hard code it?

Input: None

Output: Write a program that displays the pattern you see below.

Exact Output Expected:

```
ACEGIKMOQSUWY
CEGIKMOQSUWY
EGIKMOQSUWY
GIKMOQSUWY
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
EGIKMOQSUWY
GIKMOQSUWY
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
IKMOQSUWY
KMOQSUWY
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
MOQSUWY
OQSUWY
QSUWY
SUWY
UWY
WY
Y
QSUWY
SUWY
UWY
WY
Y
UWY
WY
Y
Y
```

2. Barb

Program Name: Barb.java **Input File: barb.dat**

Barb, a math teacher, has decided to supply all her students with a chart of simple math formulas and values to help them with their work. She only wants to provide a chart up to a certain number.

Each row of this chart will include a number, the square of the number, 3 times the number, and half of the number (no fractional part, just the whole number answer). The chart will start at 1 and continue up to a certain number Barb decides. Each chart value will be separated by exactly 5 spaces. The first line will be the chart heading which will have exactly 3 spaces between each item in the heading.

```
A   A*A   3*A   A/2
```

All numbers will be displayed as integers.

Input: A single integer N (0 < N < 100).

Output: A math chart as shown in the example below and formatted exactly as described above.

Sample Input:

```
10
```

Sample Output:

```
A   A*A   3*A   A/2
1     1     3     0
2     4     6     1
3     9     9     1
4     16     12     2
5     25     15     2
6     36     18     3
7     49     21     3
8     64     24     4
9     81     27     4
10     100     30     5
```

3. Bryce

Program Name: Bryce.java **Input File: bryce.dat**

Bryce has taken an interest in Roman numerals beyond those appearing on some clock, watch face or sundial. The Olympics and the NFL Super Bowl are probably the most visible users of Roman numerals, but they also occur in books as chapter numbers and preface page numbers, movie titles and years of production, building cornerstones with the year of construction, generational name suffixes like James Bond IV, and more.

Bryce has asked your team to write a program to help him check his practice work. He starts with whole numbers and then tries to write the equivalent Roman numerals. There have been numerous variations throughout the ages, but the modern interpretation is limited to the following symbols and rules.

Symbol	M	D	C	L	X	V	I
Value	1000	500	100	50	10	5	1

The basic rule is that the symbols appear in order of descending values and, in the simplest form, values of symbols are added together so MMXVIII = 1000 + 1000 + 10 + 5 + 1 + 1 + 1 = 2018.

To keep the length of symbols as short as possible, DD is always just M, CCCCC is always just D, LL is always just C, XXXXX is always just L, VV is always just X, and IIIII is always just V. Only M, C, X, and I can be repeated.

The subtraction rule also shortens the length of some symbol combinations and occurs when a symbol of lower value appears before a symbol of higher value. However, they are allowed in limited situations:

- CM = 1000 – 100 = 900 replaces DCCCC for 900 and then CM cannot be followed by any more Cs or a D.
- XC = 100 – 10 = 90 replaces LXXXX for 90 and then XC cannot be followed by any more Xs or an L.
- IX = 10 – 1 = 9 replaces VIIII and then IX cannot be followed by any more Is or a V.
- IV = 5 – 1 = 4 replaces IIII and IV cannot be followed by any more Is.

Input: An unknown number of positive integers with values < 5000, each on a separate line.

Output: For each input value, display a single line containing the number followed by a colon (:) and then the resulting Roman numeral with no other spacing or extraneous characters.

Sample input:
```
2018
1776
1492
1984
2649
9
```

Sample output:
```
2018:MMXVIII
1776:MDCCLXXVI
1492:MCCCCXCII
1984:MCMLXXXIV
2649:MMDCXXXXIX
9:IX
```

4. Eduardo

Program Name: Eduardo.java **Input File: eduardo.dat**

Eduardo and his girlfriend Maria have a problem. Each of their parents have access to their phones! This severely limits what they can say when they text one another. So, to confound this terrible breach of privacy they have come up with a code for their texts. Each letter in each text will be shifted to the right within the alphabet by a certain number of letters. The shift number will be the length of the first word in the text. If the shift goes past the letter z, then the code wraps around to the beginning of the alphabet.

To keep things simple Eduardo and Maria only text using lower case letters and never use punctuation. Spaces between words are left in the text and are not coded. If a text reads **"my mother says we cant go"** then the coded text would be **"oa oqvjgt ucau yg ecpv iq"**. Eduardo and Maria need you to write an app to code their texts before they are sent.

Input: An unknown number of texts where each complete text is on a separate line and each text contains at least one word. No word within any of the texts will contain more than 26 letters.

Output: Each complete coded text on a separate line.

Sample input:
```
my mother says we cant go
that is so unfair
i know im sorry
```

Sample output:
```
oa oqvjgt ucau yg ecpv iq
xlex mw ws yrjemv
j lopx jn tpssz
```

5. Goro

Program Name: Goro.java **Input File: goro.dat**

Goro has just learned all about the mathematical constant PI in math class. His teacher, Mr. Snigglefritz, has shown several methods to approximate PI using various formulas. Old Mr. Sniggs has given the class a very tedious assignment. He has asked they each choose a method of approximating PI and then write down the intermittent values for each step of the formula. Goro has chosen Mādhava of Sangamagrāma's formula, which looks like this:

$$\pi = \sqrt{12}\left(\frac{1}{1 \cdot 3^0} - \frac{1}{3 \cdot 3^1} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \cdots\right)$$

Well, there is no way Goro is going to do all that work by hand! He plans to write a program to print the estimated value of the expression as each subsequent term in the formula is included. What would the output of Goro's program look like?

Input: A single positive integer N representing the number of terms to use in the estimation of PI.

Output: The estimation of the value of PI after the inclusion of each term of the formula. There should be N output values, each on a separate line, rounded to five decimal places.

Sample input:
```
4
```

Sample output:
```
3.46410
3.07920
3.15618
3.13785
```

6. Harmony

Program Name: Harmony.java Input File: harmony.dat

Manually entering a long series of digits like a credit card number or other account number is prone to simple keying errors and transposing digits. Harmony recently learned about an interesting technique used to test for those types of simple data entry errors. She researched several check digit algorithms and designed a modified algorithm based on her findings. She now needs your help coding and testing it. Here is her proposed algorithm:

- Start with an account number like: 3125600196431014
- Left-most digit is in position 1 and position numbers increase to the right

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Digit	**3**	**1**	**2**	**5**	**6**	**0**	**0**	**1**	**9**	**6**	**4**	**3**	**1**	**0**	**1**	**4**

- Last digit is the check digit and it is not included in the following computations: 4
- Sum digits in odd positions: 3 + 2 + 6 + 0 + 9 + 4 + 1 + 1 = 26
- Sum digits in even positions: 1 + 5 + 0 + 1 + 6 + 3 + 0 = 16
- Count number of digits greater than 4 in odd positions: 2 (positions 5 and 9)
- Check sum is: odd_sum * 3 + even_sum + high_digit_count = 26 * 3 + 16 + 2 = 96
- Computed check digit is: 10 – rightmost digit of the check sum: 10 – 6 = 4
- If the rightmost digit of the check sum is 0, the computed check digit is also a 0

For the above example, the computed check digit (4) matches the last digit in the account number (4) and therefore the account number is VALID.

In a similar fashion, for the account number 44089320029134967, the odd sum = 36, even sum = 28, high digit count = 3 with check sum = 139 to produce a computed check digit of 10 – 9 = 1. This account number is INVALID since 7 ≠ 1.

Input: The first line of the data file contains a count of account numbers that will follow, with a single account number on each line. The account numbers consist of only digits. There will be no spaces, dashes, or any other non-digit characters. Account numbers could contain as many as 20 digits.

Output: One line for each account number containing the account number followed by a single space and either "VALID" or "INVALID" and, when invalid, follow with a single space and the correct check digit.

Sample input:
```
6
3125600196431014
44089320029134967
4408932002913490
354102320048
42018240304918579385
6960984005
```

Sample output:
```
3125600196431014 VALID
44089320029134967 INVALID 1
4408932002913490 INVALID 7
354102320048 VALID
42018240304918579385 INVALID 6
6960984005 VALID
```

7. Hyun

Program Name: Hyun.java **Input File: hyun.dat**

Hyun is learning to type. To help him out, you give him some sentences and have him type them out. Hyun is better at typing at some rows of the keyboard than others, so not all sentences are equally easy for him.

Hyun is typing on a standard US QWERTY keyboard. For the purposes of this problem, such keyboards have 4 rows, as shown in the figure below:

```
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
 a s d f g h j k l
  z x c v b n m
```

Each row has its own name. From the top down, they are "NUMBER", "TOP", “HOME", and "BOTTOM".

Sentences only consist of lowercase letters (a-z), numbers (0-9), and spaces. Hyun wants to know which keyboard rows a given sentence uses (ignoring the space bar). Write a program that helps him do that.

Input: Input starts with a line containing an integer T indicating the number of sentences Hyun has to type (at most 20). Following this are T lines, each containing one sentence. A sentence is a non-empty string of at most 100 characters that are either lowercase letters, digits, or spaces.

Output: For each sentence, print the test case number followed by a space separated list of keyboard rows needed to type the sentence. The needed rows should be printed from top to bottom, formatted as in the samples.

Sample input:

```
7
the
quick
brown
fox
jumps over
the lazy dog
10 times
```

Sample output:

```
Case #1: TOP HOME
Case #2: TOP HOME BOTTOM
Case #3: TOP BOTTOM
Case #4: TOP HOME BOTTOM
Case #5: TOP HOME BOTTOM
Case #6: TOP HOME BOTTOM
Case #7: NUMBER TOP HOME BOTTOM
```

8. Joaquin

Program Name: Joaquin.java Input File: joaquin.dat

Joaquin loves shapes and squares and wants to write a program to make these designs. For the values 10 and 11, the resulting **"squares within a square"** patterns would be as shown below. Can you help him?

```
XXXXXXXXXX              XXXXXXXXXXX
X        X              X         X
X XXXXXX X              X XXXXXXX X
X X    X X              X X     X X
X X XX X X              X X XXX X X
X X XX X X              X X X X X X
X X    X X              X X XXX X X
X XXXXXX X              X X     X X
X        X              X XXXXXXX X
XXXXXXXXXX              X         X
                        XXXXXXXXXXX
```

Input: The first line is T (T ≤ 100), the number of data sets. Each data set has one integer >1 and ≤ 20.

Output: A **"squares within a square"** of uppercase X's as shown above, each pattern followed by a row of 20 dashes.

Example Input:
```
3
10
11
5
```

Example Output:
```
XXXXXXXXXX
X        X
X XXXXXX X
X X    X X
X X XX X X
X X XX X X
X X    X X
X XXXXXX X
X        X
XXXXXXXXXX
--------------------
XXXXXXXXXXX
X         X
X XXXXXXX X
X X     X X
X X XXX X X
X X X X X X
X X XXX X X
X X     X X
X XXXXXXX X
X         X
XXXXXXXXXXX
--------------------
XXXXX
X   X
X X X
X   X
XXXXX
--------------------
```

9. Mary

Program Name: Mary.java **Input File: mary.dat**

Mary just got a new toy! The toy has several vertical wheels spinning in the air, and each wheel can contain the letters of a word written on it. With one button press, she can spin the wheel either forwards or backwards through the letters of the word, displaying a different letter of the word at the top of the wheel. There are so many words she can create, some making sense, others just nonsense! Spinning backwards from the first character moves to the last character, and spinning forwards from the last character goes to the first character. Initially, all wheels are set to their first character displayed at the top.

Mary also recently learned about palindromes, words spelled the same written forwards and backwards. For example, "RACECAR" and "BOB" are both palindromes, while "BUGGY" and "ALICE" are not.

As she loves both her new toy and palindromes, she wants to change the display formed by the top letters of all the wheels to form a palindromic word, using as few button presses as possible. For example, if three wheels contain the words “BING”, “OOPS” and “BANG”, the top letters being ‘B’, ‘O’, and ‘B’, she has a palindrome already and needs no button pushes on any of the wheels to find one.

If Mary includes a fourth wheel and uses the words “DARTS”, “MOON, “SAM” and “BLAH”, she sees that “DMSB”, the current word formed by the top letters of each wheel, is certainly not a palindrome, but with a single push forward on the first wheel the letter ‘A’ is now at the top, no push on the second wheel keeping ‘M’ at the top, a reverse push on the third wheel to see another ‘M’ at the top, and two pushes either way on the fourth wheel to reveal another ‘A’ creates the palindrome “AMMA”.

For five wheels with words “ABC”, “DEFG”, “HIJ”, “KL”, and “MNOP”, she sees no matter how many pushes on any wheels, it is impossible to form a palindrome.

Input: The first line is T ($0 < T \leq 15$), the number of data sets to follow. The first line of each data set has N ($0 < N \leq 25$), the number of wheels being used on Mary’s toy, each with a word on it. The next N lines each have a string of uppercase characters, the characters on that wheel. The wheel is currently pointing to the character in the top position. There are at most 10 characters on each wheel.

Output: For each case, print the palindrome that requires the fewest button presses to make, a space, and the number of presses needed to make it. If it is impossible to make any palindrome, print "IMPOSSIBLE". If there are many palindromes that can be made with the same cost, print the one that comes first lexicographically.

Sample input:
```
3
3
BING
OOPS
BANG
4
DARTS
MOON
SAM
BLAH
5
ABC
DEFG
HIJ
KL
MNOP
```

Sample output:
```
BOB 0
AMMA 4
IMPOSSIBLE
```

10. Nisha

Program Name: Nisha.java **Input File: nisha.dat**

Nisha just learned about writing numbers in binary in class. She loves powers of two, and often doodles during class. Her most recent doodles are lists of binary representations of numbers. Nisha doesn't consider a list completed until she's written out every number from 1 to 2^k (inclusive) for some positive integer k. For example, here's her list for k =3.

```
1
10
11
100
101
110
111
1000
```

Nisha was curious how many times she writes the digit 1 as part of this list. Help her find the answer by writing a program.

Input: The first line is T (at most 20), the number of test cases to follow. Each test case contains a single positive integer k (at most 50). This means that Nisha's list contains the binary representations of all numbers between 1 and 2^k (inclusive).

Output: For each test case, print the case number and the total number of times the number 1 appears in the binary expansion of all integers between 1 and 2^k (inclusive), formatted as in the sample. Note the answer may not fit into a 32-bit integer.

Sample input:
```
3
3
1
15
```

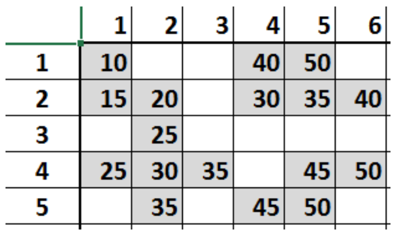
Sample output:
```
Case #1: 13
Case #2: 2
Case #3: 245761
```

11. Riley

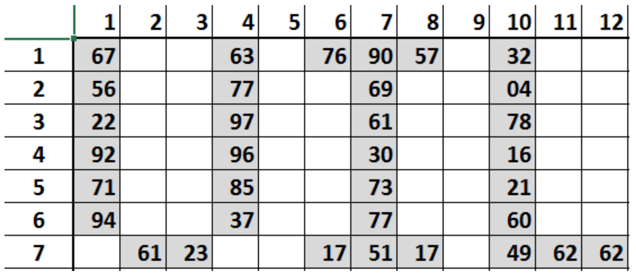
Program Name: Riley.java **Input File: riley.dat**

Riley had cousins and friends that evacuated their homes this past summer because of wildfires. She understands that many factors like terrain, humidity, and wind affect the growth and advancement of wildfires and is interested in the idea of calculating the size of wild fires across large areas. After studying computer science for UIL, she wonders how accurate it might be to overlay a map with a north-south by east-west grid to calculate the total fire coverage of a particular area and also identify clusters of fire activity within that area. Each individual square of the grid represents a 25-acre section of land and is equivalent to about 19 football fields. That's a lot of football!

Each of the individual grid squares has a fire-coverage rating between 0% and 100%. A fire cluster is a line of squares, an area of squares, or a ring of squares that touch one another on one of the sides or on a corner. A border of squares that are free of fire completely surround each cluster. It is possible to have a ring of fire with a fire-free area in the middle. The border around the edge of the defined area is free of fire. The images below show clusters of touching cells as expressed in the sample data.



	1	2	3	4	5	6
1	10			40	50	
2	15	20		30	35	40
3		25				
4	25	30	35		45	50
5		35		45	50	



	1	2	3	4	5	6	7	8	9	10	11	12
1	67			63		76	90	57		32		
2	56			77			69			04		
3	22			97			61			78		
4	92			96			30			16		
5	71			85			73			21		
6	94			37			77			60		
7		61	23			17	51	17		49	62	62

Riley asked your team to design and write a program to find the total fire coverage across an area and produce a list of the individual clusters that contain some coverage by wild fire.

Input: The first line of the data file contains a count of the number of data sets. The first line of each data set contains two positive integers that are between 2 and 100, the number of rows R and columns C in the grid. The next R rows will each contain exactly C integers with values between 0 and 100, separated by one or more spaces. The values provide the percentages of fire coverage in the squares 1…C of that particular row.

Output: For each data set, report the total fire coverage in acres for that area and display a list of individual clusters of fire sorted by descending size, also in acres. Acreage values must be rounded to 2 decimal places. The output for each data set is followed by a line containing "============".

Sample input:
```
2
5 6
10  0  0 40 50  0
15 20  0 30 35 40
 0 25  0  0  0  0
25 30 35  0 45 50
 0 35  0 45 50  0
7 12
67  0  0 63  0 76 90 57  0 32  0  0
56  0  0 77  0  0 69  0  0 04  0  0
22  0  0 97  0  0 61  0  0 78  0  0
92  0  0 96  0  0 30  0  0 16  0  0
71  0  0 85  0  0 73  0  0 21  0  0
94  0  0 37  0  0 77  0  0 60  0  0
 0 61 23  0  0 17 51 17  0 49 62 62
```

Sample output:
```
145.00 acres
96.25 acres
48.75 acres
============
485.75 acres
235.25 acres
154.50 acres
96.00 acres
============
```

12. Sara

Program Name: Sara.java **Input File: sara.dat**

Sara is a wedding planner. For each wedding Sara's clients provide a list of guests from both the groom and the bride. Usually those lists overlap and have a few of the same names on both lists. For each wedding, seating arrangements are very important. Sara needs four alphabetized lists of guests for each wedding, the first of all guests attending, another of guests appearing on both bride’s and groom’s lists (they can be seated next to anyone), a third of the bride's guests not on the grooms list (they will be seated together) and finally a list of the groom's guests not on the bride’s list (who will also be seated together). Write a program that will help Sara create her wedding day guest lists.

Input: A whole number N representing the number of weddings Sara will be planning. For each of the N weddings there will be two lines of data, each containing a list of names. The first line will be the bride's guest list and the second line will be the groom's guest list. Both the bride and groom will always have at least one guest. The names will all only be one name and there will not be any duplicates within a list.

Output: The output for each wedding will consist of four lines where each line is one of the lists described above. Each list should be shown in the order and labeled as shown in the sample output. If any list is empty, print the label for that list followed by "none". End each set of wedding lists with a row of 20 dashes.

Sample input:
```
2
Aaron Kathryn Joanna Dylan Dasha Tamas Anu Vlad
Lucia Mary Boris David Aaron Joanna Masha Goro
Botan Dawn Amanda Ian Gowri
John Katya Anusha Ian Dawn Sarah Nishant Mateo
```

Sample output:
```
Guests: Aaron Anu Boris Dasha David Dylan Goro Joanna Kathryn Lucia Mary Masha Tamas Vlad
Guests of Both: Aaron Joanna
Bride's Guests: Anu Dasha Dylan Kathryn Tamas Vlad
Groom's Guests: Boris David Goro Lucia Mary Masha
--------------------
Guests: Amanda Anusha Botan Dawn Gowri Ian John Katya Mateo Nishant Sarah
Guests of Both: Dawn Ian
Bride's Guests: Amanda Botan Gowri
Groom's Guests: Anusha John Katya Mateo Nishant Sarah
--------------------
```