TXCSOpen Programming Problem Set

UIL District 2020 + Custom Problems

IMPORTANT: Use Standard System input and NOT file input (System.in, cin)

I. General Notes

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

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

UIL District

1. Abril
2. Brittany
3. Emmanuel
4. Guowei
5. Ina
6. Josefa
7. Kenneth
8. Magdalena
9. Noah
10. Ramiro
11. Seema
12. Wojtek

Custom

13. Least Least Common Multiple Sum
14. Constellations
15. Power Walking
16. A Long Piece of String
17. Really Mean Question
18. Pattern Finding

1. Abril

Program Name: Abril.java **Input File: none**

Abril likes to create ASCII text art and has challenged your team to produce the output shown below.

Input: There is no input for this program.

Output: The exact output shown below with the last line at the leftmost edge of the console screen.

Sample input: None

Sample output:

```
    UIL 2020
  Java Program
    Launch!
================
        ^
        ^
        |
       | |
      |   |
     |_____|
     |     |
     |     |
     |  U  |
     |  I  |
     |  L  |
     |     |
     |  T  |
     |  X  |
     |     |
     |     |
     [-----]
      |###|
       |#|
        #
       <#>
      (###)
     {~~~~~}
    {~~} {~~}
   {~~}   {~~}
  {~~~~} {~~~~}
{~~~~~} {~~~~~}
```

2. Brittany

Program Name: Brittany.java **Input File: brittany.dat**

Brittany really likes math and completely understands how to work with averages. She has wondered how difficult it would be do write a program to compute and display the sum and average of a series of numbers. She has asked your UIL programming team to help her write such a program.

Input: A list of N positive decimal values, with $5 \leq N \leq 15$ and separated by single spaces, all on one line. Each of the values are < 1000.

Output: The sum and average of the values rounded to 2 digits after decimal points with one space between them.

Sample input:

```
132.30 80.17 434.99 962.68 754.54 31.36 603.81 816.73 36.41 93.45 51.71
```

Sample output:

```
3998.15 363.47
```

3. Emmanuel

Program Name: Emmanuel.java **Input File: emmanuel.dat**

Emmanuel is in the same statistics course as Brianna, and upon hearing she wrote a program to calculate the range of a set of data, he wanted to write a program to calculate the mode of a set of data. The mode of a set of data is defined as the value that appears the most often. Can you help Emmanuel write a program to calculate the mode as well as the number of occurrences of the mode?

Input: Input starts with a line containing an integer N (1<=N<=10), the number of test cases. The following N lines will contain a set of positive integers between 1 and 100 (inclusive) inside of curly braces. Data is comma separated with no spaces. The set is allowed to have duplicates and will have at least one element and no more than 100 elements. The set of data is guaranteed to have only one mode. In the below sample input, the 5th sample is a single line of text in the data file.

Output: For each test case, output the mode for the set of data as well as the number of occurrences of that mode formatted as shown below.

Sample Input:

```
5
{6,2,8,4,5,4,3,5,7,11,12,100,3,4,5,3,4,66}
{2,2}
{23,15,3,4,15,2}
{4}
{96,40,42,30,47,7,61,88,36,87,94,60,33,84,62,83,27,80,51,95,63,4
9,66,24,97,93,38,15,81,64,44,75,76,31,79,54,41,32,70,57,20,11,73
,72,50,39,86,5,34,99,37,13,23,55,35,45,43,77,65,12,82,3,28,46,53
,6,17,29,9,25,69,67,26,78,48,19,89,14,10,4,91,85,8,56,21,99,16,7
1,18,90,22,52,68,92,58,59}
```

Sample Output:

```
4 appeared 4 time(s)
2 appeared 2 time(s)
15 appeared 2 time(s)
4 appeared 1 time(s)
99 appeared 2 time(s)
```

4. Guowei

Program Name: Guowei.java **Input File: guowei.dat**

Guowei has gotten a job working for the local television station KUIL. His first job at the station is to be responsible for the news ticker. The news ticker is the crawler at the bottom of the screen that has breaking news, program alerts, public service announcements or sports scores. The station can display 40 characters at a time across the bottom of the screen and they scroll past from right to left. Characters appear on the right and disappear on the left and when the ticker first appears it contains 40 characters. The message wraps around within the crawler. That means that if enough characters are scrolled across the screen the original characters will reappear on the right. Write a program that will help Guowei create a news ticker.

Input: A single value N (N <= 20) followed by N sets of data consisting of two lines each. The first line of each data set will be a single value C (1 <= C <= 160) indicating the number of characters that have scrolled off the screen. The second line of data will be a line of text to be displayed in the ticker. This line of text will always contain more than 40 characters (this includes spaces and punctuation).

Output: The 40 characters that will be displayed within the ticker after C characters have been scrolled across the screen. If the first character in the ticker wraps around to be displayed again, there must be a space between the last character and the first character.

Sample input:

```
3
5
All Tom Green County schools will have a 10 o'clock start
tomorrow morning.
25
NEWS FLASH! Mayor Bob Baggins has announced his campaign for
Texas State Senate seat District 19.
67
Don't miss the Children's Miracle Network Telethon.
```

Sample output:

```
om Green County schools will have a 10 o
gins has announced his campaign for Texa
Children's Miracle Network Telethon. Don
```

5. Ina

Program Name: Ina.java **Input File: ina.dat**

The Universal Internet Lab (UIL) recently developed an ingenious new method of connecting its computers. Each computer is directly connected to some other set of computers such that there is exactly one sequence of connections between any pair of computers. Stated differently, by treating the computers as vertices and connections as edges, the computers and their connections form a tree.

While this may seem constraining, connecting computers in this manner has some benefits. For example, software updates can now be installed in a distributed manner. Instead of each computer connecting to some central server to download a software update, only one computer needs to be updated. Then, by using these connections between computers, one computer can update all the other computers directly or indirectly!

Here's how it works: consider a connection where one end of the connection has a computer with the update (hereafter referred to as the source) and a computer without the update (hereafter referred to as the sink). Then in L seconds, the sink can download and install the update from the source. Each sink can only download from a single source at a time, and each source can only be accessed by a single sink at a time.

At the beginning, only computer S has the update. If Ina chooses the optimal ordering for updates, what is the fastest that all computers can be updated?

Input: The first line of input is an integer T (1 <= T <= 50), the number of test cases. Each test case starts with three space-separated integers N (1 <= N <= 100), S (1 <= S <= N) and L (1 <= L <= 1,000,000), the number of computers, the computer which initially has the update, and the amount of time it takes to transfer a single update. Then follow N - 1 lines, each with two space-separated integers U V (1 <= U, V <= N): the two computers connected by that connection. It is guaranteed that the set of edges forms a tree.

Output: For each test case, output a single integer: the minimum amount of time needed to update all computers. Format the output as in the samples.

Sample Description: In the first test case, an optimal solution is:

Computer 3 downloads from computer 1 (2 seconds)
Computer 4 downloads from computer 3 (2 seconds)
Simultaneously, computer 5 downloads from computer 3 and computer 2 downloads from computer 1 (2 seconds).

Thus the whole process takes 6 seconds. The final two updates can be done simultaneously because no computer is involved in multiple updates, and in each update one computer has the update and one does not.
(Sample input and output on next page)

Ina (cont.)

Sample input:

```
3
5 1 2
1 2
1 3
3 4
3 5
3 2 4
1 2
3 2
5 5 3
5 3
2 1
1 3
2 4
```

Sample output:

```
Case #1: 6
Case #2: 8
Case #3: 12
```

6. Josefa

Program Name: Josefa.java **Input File: josefa.dat**

Josefa is on her high school's UIL computer science team. At this week's practice coach Snigglefritz taught the team how to convert a decimal value to an eight bit two's complement binary number. The process is easy enough but Josefa would like a program to help double check all of the practice problems coach has assigned. She would like to enter a decimal whole number, either positive or negative, and see the equivalent two's complement binary number so she can see if that matches the number she has calculated by hand.

Input: A number N that represents the number of whole numbers to be converted to two's complement binary numbers. N whole numbers X each on a separate line where -128 <= X <= 127.

Output: Each X and its eight bit two's complement equivalent separated by a space, =, and then another space. Each pair should be on a separate line.

Sample input:

```
4
15
-16
-49
72
```

Sample output:

```
15 = 00001111
-16 = 11110000
-49 = 11001111
72 = 01001000
```

7. Kenneth

Program Name: Kenneth.java **Input File: kenneth.dat**

Kenneth likes to solve Sudoku puzzles. Sudoku puzzles are not Math puzzles, they are logic puzzles and are a great way to sharpen your problem solving skills. The puzzle consists of a 9 × 9 grid with 9 rows and 9 columns and each cell of the grid will ultimately contain a single digit 1…9. The puzzle starts with some of the cells containing digits and some of the cells empty like the puzzle shown below on the left. The challenge is to fill the remaining cells correctly so that each row and column contains each of the digits 1…9 exactly once, there are no duplicate cells. In addition, each 3 × 3 sub-grid working across and down the puzzle must also contain all of the digits 1…9 with no duplicates. The shading in the following puzzles shows the 3 × 3 sub-grids. The grid on the left is the starting puzzle and the grid on the right is the proposed solution. The following example is a correct solution.

Puzzle

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

Bold digits are start of puzzle

Solution

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

Row numbers and column numbers are simply 1…9 with row and column 1 in the top left corner of the grid and row and column 9 in the bottom right corner.

The following puzzle and proposed solution contains errors. It has a 2 instead of a 5 in row 2 column 3 and an 8 instead of a 6 in row 7 column 5 causing both row and column errors. Those errors also cause the following sub-grids to contain errors: top-left and bottom-center.

Puzzle

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

Bold digits are start of puzzle

Solution

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

Kenneth is confident in his skills but has asked your UIL programming team for assistance creating a program that confirms whether or not a proposed solution is correct. The program does NOT solve the puzzle.

Input: First line will contain a number $1 \leq T \leq 10$ as the number of test cases. Each test case will be followed by nine lines which are the rows and each row contains exactly 9 digits separated by spaces with all digits being 1…9 and no stray or extra characters.

Output: For each test case, the first line of output starts with "`GRID #t:`" and a space with `t` as the test case number starting with 1 and that is followed by either "`SOLUTION IS CORRECT`" or "`NOT A SOLUTION`". When the proposed solution is correct, no other details are output. When the proposed solution is not correct, two more lines will be output. The first detail line starts with "`>> ROWS WITH ERRORS:`" and a space and either a list of row numbers containing errors or "`NONE`". The second detail line starts with "`>> COLUMNS WITH ERRORS:`" and a space and either a list of column numbers containing errors or "`NONE`". Display multiple row and column numbers in ascending order separated by single spaces. Each test case result is followed by a row containing 12 equal signs "`============`".

Sample input:

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

Sample output:

```
GRID #1: SOLUTION IS CORRECT
============
GRID #2: NOT A SOLUTION
>> ROWS WITH ERRORS: NONE
>> COLUMNS WITH ERRORS: 1 4 5 7
============
GRID #3: NOT A SOLUTION
>> ROWS WITH ERRORS: 2 7
>> COLUMNS WITH ERRORS: 3 5
============
```

8. Magdalena

Program Name: Magdalena.java **Input File: magdalena.dat**

In Magdalena's mathematics class, they sometimes work with large bases. To handle numbers in such bases, the class has agreed on the following convention: the first ten digits are the Arabic numerals, (0, 1, ... 9), the next twenty-six digits are the upper-case Latin alphabet (A, B, C, ..., Z), and the next twenty-six digits, are the lowercase Latin alphabet (a, b, c, ..., z). With this schema, the students in the class can work in any base from 2 to 62 (both ends inclusive).

For example, Magdalena can convert 314159265 in base 10 to 1B8XeX in base 48. To prove this to herself, she double checks that:

$$314159265 = 1 * 48^5 + 11 * 48^4 + 8 * 48^3 + 33 * 48^2 + 40 * 48^1 + 33 * 48^0$$

Magdalena feels comfortable with doing large base conversions and is now considering new challenges. She likes large numbers, and she also likes numbers with large digit sums.

The digit sum of a number X in base B can be computed the following way: write out the digits of X in base B, sum those digits, and output the result in base B. For example, the digit sum of 314159265 in base 10 is 36. The digit sum of 1B8XeX in base 48 is 2U (126 in base 10).

Now Magdalena wonders: given a large number X in some base B, what is the largest digit sum over all numbers Y <= X in base B?

Input: The first line of input is an integer T (1 <= T <= 50), the number of test cases. Each test case has two space-separated tokens. The first token is B (2 <= B <= 62), an integer written in base 10 representing the base of X. The second token is positive integer X, written in base B. X will have no more than 200 digits in whatever base it is written in and has no leading 0s. X will be a valid integer in base B.

Output: For each test case, output a single integer: the largest possible digit sum of any integer <= X, expressed in base B. Format the output as in the samples.

Sample Description: The digit sum of a single digit number is that digit itself. Therefore the largest digit sum of numbers <= 5 is 5.

Sample input:

```
3
10 5
16 10
48 1B8XeX
```

Sample output:

```
Case #1: 5
Case #2: F
Case #3: 4h
```

9. Noah

Program Name: Noah.java **Input File: noah.dat**

Noah is a columnist for his school's newspaper and he's writing an article on the most driven vehicles among the student body. He has already collected the data in regard to the make, model, and year of each student's individual vehicle. The problem is his data is all over the place and needs to be organized so that he can analyze it. Can you write a program that will sort all the data alphabetically by make, model, then year as well as display the breakdown of each vehicle by make, model and year?

Input: Input starts with a line containing an integer N (1<=N<=40), the number of students who drive a car at Noah's school. The following N lines will contain the make, model, year for each vehicle Noah collected data for. The make, model, year will be separated by a comma.

Output: The output for this program is broken into four portions:

1) Data Sorted
2) Make Breakdown
3) Model Breakdown
4) Year Breakdown

The Data Sorted portion is displayed alphabetically by make, with the tiebreaker being model. If there is then a tie between both make and model, the final tiebreaker is year which is displayed from lowest to highest.

For the make, model, and year breakdowns, the output is the number of occurrences of each. It is important to note that each breakdown is sorted. Make and model are sorted alphabetically, and year is sorted numerically from lowest to highest.

Sample Input:

```
16
Chevy,Cruze,2019
Chevy,Silverado,2006
Ford,F150,2010
Dodge,Durango,2020
Dodge,Ram1500,2008
Chevy,Silverado,2006
Ford,F150,2015
Chevy,Cruze,2009
Ford,Explorer,2006
Dodge,Ram1500,2018
Dodge,Ram1500,2019
Ford,F150,2015
Ford,Mustang,2017
Ford,Edge,2012
Dodge,Charger,2017
Dodge,Challenger,2012
```

(sample output on next page)

Sample Output: (Noah)

```
---Data Sorted---
Chevy Cruze 2009
Chevy Cruze 2019
Chevy Silverado 2006
Chevy Silverado 2006
Dodge Challenger 2012
Dodge Charger 2017
Dodge Durango 2020
Dodge Ram1500 2008
Dodge Ram1500 2018
Dodge Ram1500 2019
Ford Edge 2012
Ford Explorer 2006
Ford F150 2010
Ford F150 2015
Ford F150 2015
Ford Mustang 2017

---Make Breakdown---
Chevy: 4
Dodge: 6
Ford: 6

---Model Breakdown---
Challenger: 1
Charger: 1
Cruze: 2
Durango: 1
Edge: 1
Explorer: 1
F150: 3
Mustang: 1
Ram1500: 3
Silverado: 2

---Year Breakdown---
2006: 3
2008: 1
2009: 1
2010: 1
2012: 2
2015: 2
2017: 2
2018: 1
2019: 2
2020: 1
```

10. Ramiro

Program Name: Ramiro.java **Input File: ramiro.dat**

In 1950, Richard Hamming introduced the concept of Hamming distance. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

Ramiro needs your help in computing the Hamming distance between hexadecimal strings. Recall, that each hex digit can be represented as four bits. For example, A in base 16 is equivalent to 1010 in base 2 and 8 in base 16 is equivalent to 1000 in base 2. The resulting Hamming distance between the two hex strings A and 8 would be 1.

1010
1000
✓✓x✓

Your task is to determine how many different bits are present, given the two hexadecimal strings of length 8.

Input: Input starts with a line containing an integer N (1<=N<=10). The following N lines each contain two hexadecimal strings of length 8 separated by a space.

Output: Output the Hamming distance between each of the two hexadecimal strings.

Sample Input:

```
6
FFFFFFFF 00000000
01234567 89ABCDEF
060A08AE C01D0041
BA26F57D CD3B689F
03660000 00007390
AAAAA57D AAAAA3B6
```

Sample Output:

```
32
8
16
19
13
7
```

11. Seema

Program Name: Seema.java **Input File: seema.dat**

Seema got bored riding around in her parent's car last year, so she began counting all of the out-of-state license plates on the cars she saw. (Seema lives in Texas.) After each trip she came home and simply typed the name of each state license plate she had seen on that trip. If she saw a state more than once, she typed the name of the state as many times as she saw the plate. Now it is time to tally up the out of state license plates and print a list. Your job is to write a program that will do that for Seema.

Input: An unknown number of state names, one for each out of state license plate that Seema saw. The names will all be listed on a single line each separated by a comma.

Output: The name of each state license plate that Seema saw followed by a space and then the number of times Seema saw the plate each on a separate line. The states and counts should be listed from most frequent to least. If Seema saw a license plate the same number of times, list those states in alphabetical order.

Sample input: *(Note: All the state names will be on one line in the data file.)*

```
Oklahoma,Nevada,Oklahoma,Oklahoma,Ohio,North Dakota,Missouri,Arizona,Idaho,
Arkansas,California,Arizona,California,Ohio,Washington,California
```

Sample output:

```
California 3
Oklahoma 3
Arizona 2
Ohio 2
Arkansas 1
Idaho 1
Missouri 1
Nevada 1
North Dakota 1
Washington 1
```

12. Wojtek

Program Name: Wojtek.java **Input File: wojtek.dat**

Bored in math class as always, Wojtek has been doodling in the margins of his homework. His latest fascination has been the game of Diffy. To play, he draws a square and writes down four integers at its corners. Then, on the middle of each side, he writes the positive difference of the two numbers at the corners and connects those to form a smaller square. This process continues until he produces a square with value 0 at all four corners.

Wojtek had a marvelous proof that if you began with four non-negative integers at the corners where the largest value on any corner has absolute value M, then he would arrive at a box where every corner is 0 in at most M + 3 steps. Alas, the proof is too large for this UIL packet. To verify his claim, write a program that counts the number of steps it takes to arrive at a square with 0 at each corner.

Input: The first line of input is T (1 <= T <= 50), the number of test cases. Each test case is a single line with four space-separated integers: the starting numbers on the corners in clockwise order. The absolute value of each integer is at most 100,000.

Output: For each test case, output the number of steps it takes to get from the original square to a square where every corner is 0. Format your output as in the samples.

Sample Description: The first sample takes 6 steps from the original values. These are the values of the corners after each step:

```
6, -2, 3, 5
8, 5, 2, 1  (Step 1)
3, 3, 1, 7
0, 2, 6, 4
2, 4, 2, 4
2, 2, 2, 2
0, 0, 0, 0  (Step 6)
```

Sample input:

```
4
6 -2 3 5
3 1 4 1
2 -7 1 -8
0 1 2 3
```

Sample output:

```
Case #1: 6
Case #2: 4
Case #3: 3
Case #4: 5
```

Custom Problem 1. Least Least Common Multiple Sum

Program Name: LLCMS.java **Input: standard input (System.in)**

You hit your head on a rock and for some reason really like least common multiples now but hate large numbers. Once you get up from said rock, you hallucinate and see an array of numbers, just waiting to be divided up into at most k intervals in a way that minimizes the sum of the lcms among each interval in the array.

Input: On the first line, you will receive a number t ($1 \leq t \leq 3$), the number of test cases to follow. On the first line of each test case, you will receive the numbers n ($1 \leq n \leq 300$) and k ($1 \leq k \leq 100$), denoting the length of the array and the number of groups respectively. On the second line of each test case, you will receive space separated integers, denoting the values of the array. All values are greater than zero and less than 10^5.

Output: Output the least possible sum of the least common multiples using at most k contiguous groups.

Sample input:

```
1
8 3
1 2 2 2 3 3 1 16
```

Sample output:

```
21
```

Custom Problem 2. Constellations

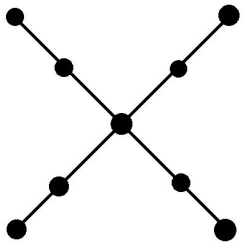
Program Name: Constellations.java **Input: standard input (System.in)**

You've spent years memorizing every single intricacy of the night sky and thus consider yourself a master stargazer. However, one day, the stars get all mixed around! As the person who was most familiar with the old night sky, it is now your job to make new constellations. The new night sky forms an undirected unweighted graph with n vertices and n-1 edges..

With your newfound role of absolute power in divining what the new constellations should be, you decide that valid constellations will form a star (like an asterisk) where the central node has degree d ($2 \leq d \leq n$) and each line from the central node to the edge of the constellation contains r ($1 \leq r \leq n$) edges.

You will need to output whether it is possible or impossible to fill the night sky (make all edges part of a constellation) with constellations that have specific values of d and r.

Sample constellation with degree 4 and radius 2 (made in ms paint):



Input: The first line will contain space separated integers n and k ($2 < n \leq 50000$; $1 \leq k \leq 500$), the number of stars and queries respectively. The next n-1 lines will contain integers a and b ($1 \leq a,b \leq n$), indicating an unweighted edge between a and b. After that, you will receive k lines containing queries for d and r.

Output: Output k lines, with the ith line containing the answer to the ith query.

Sample input:

```
17 5
1 2
2 3
3 4
4 5
3 6
6 7
3 8
8 9
```

```
9 10
8 12
12 11
8 13
8 14
14 15
8 16
16 17
4 2
2 1
5 1
2 2
3 1
```

Sample output:

```
1
1
0
1
0
```

Custom Problem 3. Power Walking

Program Name: PowerWalking.java **Input: standard input (System.in)**

You live in a very weird neighborhood. You enjoy walking in it between your two houses a and b (staying 6 feet apart from everybody else), but get tired if you walk too much. However, as the sun comes up it becomes exponentially more difficult to walk along each road. Being the procrastinator you are, you want to choose the time of the day to leave so that you put in at most e ($1 < e < 10^9$) effort into your walk.

Your neighborhood is represented by an undirected, weighted graph. The effort it takes to traverse each edge in this graph is equal to the initial weight of an edge to the power of the time you left your house. Sum the efforts of each edge along a path to find the total effort of a path.

Input: On the first line, you will receive a number t ($1 \leq t \leq 5$), indicating the t test cases to follow. For each test case you will receive three space separated integers, n, v, and e ($1 < n,v < 10^5$), indicating the number of lines to follow in the test case, the number of vertices, and the effort value respectively. Each of these n lines will have three space separated integers a, b, and w ($1 \leq w \leq 10^6$) indicating an undirected weighted path from node a to node b. Your houses are at nodes 1 and 2.

Output: For each test case you will output the latest time $t \geq 1$ that you can leave your house without expending e effort. T should be a floating point number, rounded to three decimal places. Don't be concerned about off by 0.001 errors.

Sample input:

```
1
4 5 10
1 2 20
1 3 2
3 2 3
5 4 3
```

Sample output:

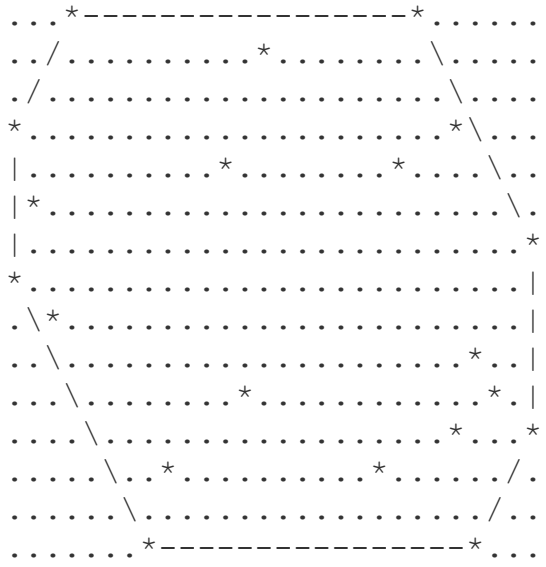
```
1.729
```

Custom Problem 4. A Long Piece of String

Program Name: LongString.java **Input: standard input (System.in)**

You have an infinitely long piece of string. You are standing on an infinitely large coordinate plane, with n ($3 \leq n \leq 50{,}000$) push pins sticking into the plane. Each of these push pins has an integral X and Y coordinate ($0 \leq X_i, Y_i \leq 1{,}000{,}000$). Your task is to take your infinitely long piece of string, cut it once into a finitely long piece of string and another infinitely long piece of string, and wrap the finite piece of string around all of the push pins in this grid. Once you are done, your string should form a convex polygon, and each push pin is either along the edge of this convex polygon or is inside the polygon.

In the figure below, the * represent push pins, and the dashed line represents a possible configuration of the string.



The push pins the string wraps around are located at (18,0), (6,-6), (0,-5), (-3,-3), (-17,0), (-7,7), (0,4), and (3,3). This yields a distance of 70.8700576850888(...). Our output will print only two decimal places, so the distance will be reported as 70.87.

Find the shortest possible length of the finitely long piece of string.

Input: Line one contains a single integer, n. Lines 2 through n+1 contain two space-separated integers, X_i and Y_i that specify the location of a push pin.

Output: You will output a single number L, the shortest possible length of the string. Output your answer to two decimal places. Do not round the length of the string; truncate it instead.

(Sample input and output on next page)

Sample Input:

```
20
2 10
3 7
22 15
12 11
20 3
28 9
1 12
9 3
14 14
25 6
8 1
25 1
28 4
24 12
4 15
13 5
26 5
21 11
24 4
1 8
```

Sample Output:

```
70.87
```

Custom Problem 5. Really Mean Question

Program Name: RMQ.java **Input: standard input (System.in)**

You are inventing a new data structure that combines the properties of a priority queue with the range indexing of an array. This data structure has one method: Select a range of the array and return the maximum value in that range, removing that value from the array. Implement it.

Input: On the first line, you will receive n and q ($1 < n,q < 10^5$), indicating the number of integers in the original array and the number of queries respectively. On the next line, you will receive a list of unique space separated integers, indicating the values in the original array. All values are greater than zero. On the subsequent q lines, you will receive two space separated integers, a and b, which represent the values that go into the data structure's method. A is inclusive, b is exclusive, the array is zero indexed, and each method call reduces the size of the array by one.

Output: You will output q lines showing the result of each query (the maximum value on [a,b) that you pop).

Sample input:

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

Sample output:

```
1
6
7
9
```

Custom Problem 6. Pattern Finding

Program Name: PatternFinding.java **Input: standard input (System.in)**

You are given an array of numbers A of length n ($1 \leq n \leq 10^5$). Your task is to find the longest subarray A' of A such that A' occurs in the array at least K times ($1 \leq K \leq N$). Note that these subarrays may potentially overlap.

Input: Line 1 contains two space separated integers, N and K. Lines 2 to n+1 contain the contents of array A, one integer per line.

Output: You will output a single integer L, the length of the longest subarray A' that satisfies the conditions above.

Sample input:

```
13 3
1
2
3
4
2
3
4
2
3
4
3
4
2
3
```

Sample output:

```
4
```