



Computer Science Competition District 2021 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	Aimi
Problem 2	Arti
Problem 3	Carol
Problem 4	Eui
Problem 5	Harish
Problem 6	Isha
Problem 7	Joyce
Problem 8	Kostya
Problem 9	Melissa
Problem 10	Pablo
Problem 11	Reka
Problem 12	Timothy

1. Aimi

Program Name: Aimi.java

Input File: None

Aimi likes to write haikus. A haiku is a short poem consisting of three lines where the first and third lines contain five syllables and the second or middle line is made up of seven syllables. Meanwhile, Aimi's computer is on its last legs and is causing all sorts of problems. Of course, Aimi wrote a few haikus to express her frustration with her computer. Write a program that will print her haikus.

Input: There is no input for this problem

Output: The three haikus shown below. Each haiku should be followed by a line containing exactly 20 dashes.

Exact Output:

```
A file that big?  
It might be very useful.  
But now it is gone.  
-----  
The web site you seek  
cannot be located but  
countless more exist.  
-----  
Chaos reigns within.  
Reflect, repent, and reboot.  
Order shall return.  
-----
```

2. Arti

Program Name: Arti.java

Input File: arti.dat

Arti has discovered that he needs to find the square root of a long list of whole numbers. Rather than working through the list by hand and finding the square root of each number using a calculator Arti has decided to write a program to find the square root of each number for him.

Input: The first line of the input file will contain a single whole number N that is the number of integers in the file. There will be no more than 50 numbers in the file. N will be followed by N whole numbers X, where $0 < X < 5000$, and each is on a separate line.

Output: N lines where each line contains the number from the file, exactly one space and, that number's square root. The square root should be rounded to three decimal places.

Sample input:

```
5
7
8
9
51
17
```

Sample output:

```
7 2.646
8 2.828
9 3.000
51 7.141
17 4.123
```

3. Carol

Program Name: Carol.java

Input File: carol.dat

The UIL’s palindrome factory that has been producing palindromes since 1913 is in a bit of a bind. With over 100 years of palindrome producing, they’re beginning to run out of ways to make new and creative palindromes. Recall, a palindrome is a word or sequence of characters that reads the same backward as forward. Carol had the bright idea to take any word, that may or may not be a palindrome, and convert it to a palindrome. For example, “UIL” can be converted into one of two palindromes: “UILIU” or “LIUIL”. Either way, two new characters must be added either to the front or the back of “UIL”. However, “Solos” is already a palindrome, and does not need any new characters added to it to make it a palindrome. Can you help Carol write a program that given any word, will output the number of characters that need to be added to convert that word into a palindrome?

Input: Input begins with an integer N ($1 \leq N \leq 20$), the number of different test cases. Each of the following N lines will contain a word or sequence of characters with length L ($1 \leq L \leq 50$). Characters will be either alphabetic or numeric. Upper and lowercase letters are to be treated as the same letter. For example, ‘a’ and ‘A’ are both the same letter.

Output: For each word or sequence of characters, you are to output the number of character(s) that must be added to convert the word or sequence of characters into a palindrome. Characters can be added to either the front or the end of the word, but characters can NOT be added to both the front and the end. Formatting should match that presented in the Sample Output below.

Sample Input:

```
8
UIL
Solos
Java
CSRocks
12333
4456789
Password123
Racecars
```

Sample Output:

```
2 character(s) need to be added to convert UIL into a palindrome.
0 character(s) need to be added to convert Solos into a palindrome.
1 character(s) need to be added to convert Java into a palindrome.
6 character(s) need to be added to convert CSRocks into a palindrome.
2 character(s) need to be added to convert 12333 into a palindrome.
5 character(s) need to be added to convert 4456789 into a palindrome.
10 character(s) need to be added to convert Password123 into a palindrome.
1 character(s) need to be added to convert Racecars into a palindrome.
```

4. Eui

Program Name: Eui.java

Input File: eui.dat

In her statistics class, Eui came across the birthday paradox, which states that if the distribution of birthdays is uniformly random, and there are just 23 people in a room, there is roughly a 50% chance that at least two people share a birthday. This seems surprising at first glance, since there are 365 days in a year, but the number of pairs of people quadratically grows as the number of people increases.

While checking to see if any of her classmates share birthdays, Eui realized that she doesn't actually know her classmates' exact birth dates. However, she knows their birth months and has a rough estimate of the date itself. Given the information she has, she wants to know the number of possible distinct birth dates.

Input: The first line of input is an integer T ($1 \leq T \leq 50$), the number of test cases. Each test case begins with a single integer N ($1 \leq N \leq 200$), the number of students in Eui's class. Each of the next N lines contains a date range: the name of the month, the beginning of the date range, a hyphen, then the end of the date range. For example,

January 1 - 3

represents the set of dates January 1st, January 2nd, and January 3rd. It is guaranteed that all dates in the input are valid in non-leap years.

Output: For each test case, output the minimum and maximum number of distinct birth dates, formatted with the case number as in the samples.

Sample Input:

```
2
3
January 1 - 3
January 1 - 3
January 1 - 3
5
February 5 - 10
February 3 - 4
March 9 - 12
March 12 - 15
March 9 - 15
```

Sample Output:

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

Sample Explanation:

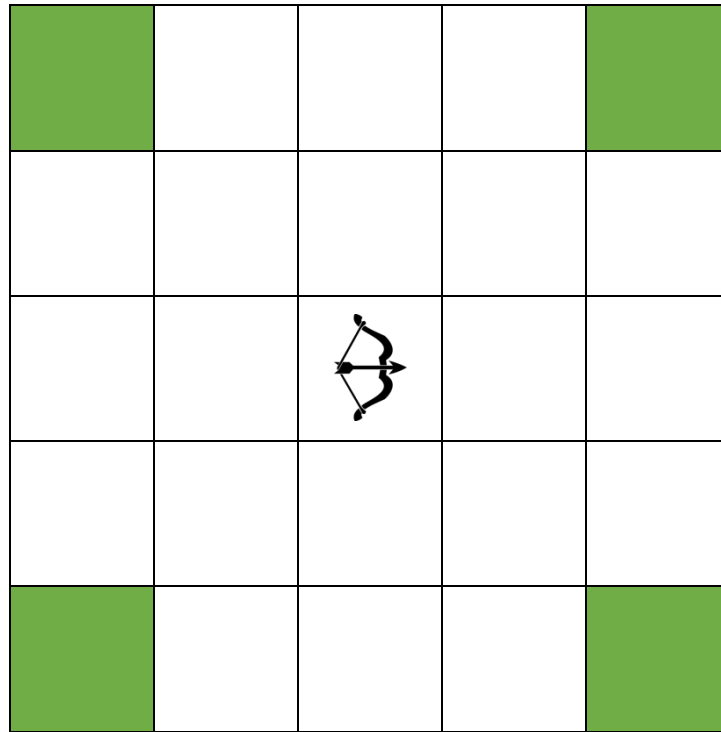
In the first sample, there are 3 students, and their birth dates are all between January 1st and January 3rd, inclusive. It is possible they all have distinct birthdays, and it is also possible that they all share a birthday.

5. Harish

Program Name: Harish.java

Input File: harish.dat

Harish is an avid Uber Intense Life board game player. Uber Intense Life, is a game similar to chess, in which players must use pieces such as the archer, squire, warrior, and blacksmith to surround the opponent's capitol and force a surrender all while protecting their own capital from enemy invasion. Harish's favorite Uber Intense Life piece is the archer, a piece known for its long-range attack. A legal attack for an archer to make is by shooting an arrow 2 units either up or down, and 2 units either left or right. An archer cannot shoot an arrow that requires the arrow to exit the board. The below figure shows an example of the four possible attacks an archer can make. The shaded squares show valid squares for the centralized archer to attack.



Harish has the idea to try and place **nine** archers on a 5x5 board such that no two archers could attack each other, i.e., no archer could fire an arrow into an already occupied square. The problem is, Harish has no way to verify if a given configuration of archers meets the above criteria. Can you help him write a program to verify that nine distinct archers are placed on a 5x5 board and that no two archers can attack each other?

Input: Input begins with an integer N ($1 \leq N \leq 20$), the number of different test cases. Each test case will consist of a 5x5 grid of characters. An "a" represents an archer's location and a "." represents an empty, unoccupied square. Each test case will be followed by a line containing ten dashes (-).

Output: For each test case, your program is to output `valid` if and only if **nine** archers are placed on the board such that no two archers could attack each other, otherwise your program should output `invalid`.

Sample input and output are on the next page to avoid a page break in the middle of input.

Harish continued:

Sample Input:

```
6
...a.
...a.
a.a..
.a.a.
a.a.a
-----
.a...
...a.
a...a
.a.a.
a...a
-----
.a.a.
....a
a...a
.a.a.
a...a
-----
.a.a.
a.a.a
.....
.....
aaa.a
-----
a.a.a
.a.a.
.....
.a...
a.a.a
-----
a..a.
.a..a
a..a.
a....
a...a
-----
```

Sample Output:

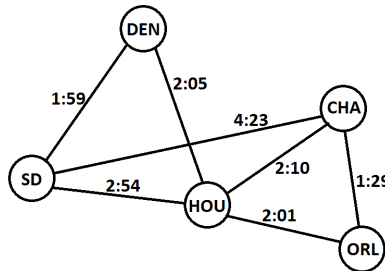
```
invalid
invalid
valid
valid
invalid
valid
```

6. Isha

Program Name: Isha.java

Input File: isha.dat

Way Cool Software Development Corporation has recently hired Isha as a junior software engineer. Their newest client is FastFlights Airlines. FastFlights is just getting started so they only fly to a few cities in the United States. Those cities are Charlotte, North Carolina, Orlando, Florida, Houston, Texas, Denver, Colorado and San Diego, California. FastFlights has asked Way Cool to create a program that will determine the arrival time for each of their flights that accounts for changing time zones. Charlotte and Orlando are in the eastern time zone. Houston is in the central time zone. Denver is in the mountain time zone and San Diego is in the pacific time zone. There is a one hour difference between each time zone. For example, when it is 11:00 A.M. in Orlando it is 8:00 A.M. in San Diego. A diagram that shows Fast Flights various flights and the flight times for each flight is shown here:



Isha has been assigned the job of writing this program. What will her program look like when it is done?

Input: A number N representing the number of flights. N lines of data each containing the departure city, departure time and destination city. A.M. and P.M. will always be capitalized and contain periods. Consider midnight as 12:00 A.M. and noon as 12:00 P.M.

Output: The departure city and local departure time followed by the destination city and the local arrival time for each flight listed in the input data. Times should be formatted such that single digit hours are shown without a leading zero but single digit minutes should be shown with a leading zero.

Sample input:

3

Houston 6:00 A.M. Orlando

Orlando 1:25 P.M. Charlotte

Charlotte 7:30 A.M. San Diego

Sample output:

Houston 6:00 A.M. Orlando 9:01 A.M.

Orlando 1:25 P.M. Charlotte 2:54 P.M.

Charlotte 7:30 A.M. San Diego 8:53 A.M.

7. Joyce

Program Name: Joyce.java

Input File: joyce.dat

A pangram is a sentence or expression that uses all the letters of the alphabet at least once. A perfect pangram is a sentence or expression that uses all the letters of the alphabet exactly once. The most well-known pangram is “The quick brown fox jumps over the lazy dog.” Notice how every letter, either uppercase or lowercase, from A-Z was used at least once in the sentence. Joyce needs your help writing a program that first determines if the sentence is a pangram or not. If the sentence is a pangram, the program needs to determine if the pangram is perfect or not. If the sentence is not a pangram, your program should output the missing letters needed to make the sentence a pangram.

Input: Input begins with an integer N ($1 \leq N \leq 20$), the number of different test cases. Each of the following N lines will contain a word or sequence of characters made up of letters, numbers, symbols, and spaces.

Output: For each test case, your program is to output “perfect pangram” if and only if all 26 letters are used exactly once in the sentence, “pangram” if all letters were used at least once or more, or “missing ...” all the letters that are missing and would be needed to make a pangram. All non-alphabetic characters are to be disregarded.

Sample Input:

```
7
The quick brown fox jumps over the lazy dog.
ZYXW, vu TSR Ponm lkj ihgfd CBA.
.,?!' " 92384 abcde FGHIJ
Mr. Jock, TV quiz PhD., bags few lynx.
My girl wove six dozen plaid jackets before she quit.
abcdefghijklmnopqrstuvwxy
AbC...
```

Sample Output:

```
pangram
missing eq
missing klmnopqrstuvwxyz
perfect pangram
pangram
perfect pangram
missing defghijklmnopqrstuvwxyz
```

8. Kostya

Program Name: Kostya.java

Input File: kostya.dat

Kostya is folding and putting away his laundry. He has a laundry bin with his N pairs of socks. Kostya pulls out individual socks from his laundry bin one at a time. Each time he pulls a sock, he pulls out any of the remaining socks in the bin with equal probability. When Kostya pulls out the first sock in a pair, he places it on his bed. When he pulls out the second sock in a pair, he folds it with the first sock (which is already on his bed) and puts the pair in a dresser.

Kostya's bed is quite small, and can only hold K individual socks at a time. If there are already K socks on the bed and Kostya pulls out the first sock in a new pair, he runs out of bed space and gets frustrated. Compute the probability that Kostya gets frustrated.

Input: The first line of input contains an integer T ($1 \leq T \leq 50$), the number of test cases. The next T lines each have two integers N K ($1 \leq K \leq N \leq 100$).

Output: For each test case, output the probability that Kostya gets frustrated while folding his laundry. Format your answer with the case number as in the samples. Round your answer to 4 decimal places.

Sample Input:

```
2
2 1
4 4
```

Sample Output:

```
Case #1: 0.6667
Case #2: 0.0000
```

Sample Explanation: In the first test case, Kostya has 2 pairs of socks and space for 1 sock on his bed. If the first two socks Kostya pulls out are from different pairs, he gets frustrated. The probability this happens is $2/3$.

In the second test case, Kostya always has enough space on his bed for all his socks, so he can never be frustrated.

9. Melissa

Program Name: Melissa.java

Input File: melissa.dat

Given any positive integer i , i can be transformed by multiplying all of its non-zero digits. If this process is repeated enough times, the integer will eventually be transformed into a single digit d in the range $[1,9]$. For example, consider the integer 62032. Multiplying $6*2*3*2$ gives the product of 72 (zero is not considered in the product as only non-zero digits are used). Multiplying $7*2$ gives the product of 14. Multiplying $1*4$ gives the product 4. 4 is in the range of $[1,9]$. Melissa needs your help writing a program, that given any positive integer will output the single digit that number is transformed to, given the transformation process described above.

Input: Input begins with an integer N ($1 \leq N \leq 20$), the number of different test cases. Each of the following N lines will contain a single integer i ($1 \leq i \leq 100000$).

Output: For each test case, your program is to output $i \rightarrow d$, where i is the given input integer and d is the single digit i is transformed into.

Sample Input:

```
7
62032
808
20
9
1023456
99999
10
```

Sample Output:

```
62032 -> 4
808 -> 8
20 -> 2
9 -> 9
1023456 -> 4
99999 -> 2
10 -> 1
```

10. Pablo

Program Name: Pablo.java

Input File: pablo.dat

Pablo is developing an app for runners, walkers or travelers that will track how far a user has run, walked or traveled. He is starting out by developing a prototype that will simply determine the distance between where the user started recording their activity and where they stop. The formula to calculate the distance between two locations determined by their GPS coordinates is the Haversine formula:

Haversine:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

With:

$$c = 2 \cdot \arctan2(\sqrt{a}, \sqrt{1-a})$$

and:

$$d = R \cdot c$$

Where:

ϕ = latitude

λ = longitude

$R = 6371$ (Earth's mean radius in km)

The units for latitude and longitude are degrees. $\Delta\phi$ = the difference between the latitudes. $\Delta\lambda$ = the difference between the longitudes.

$$\sin^2\left(\frac{\Delta\phi}{2}\right) = \sin\left(\frac{\Delta\phi}{2}\right) * \sin\left(\frac{\Delta\phi}{2}\right)$$

Pablo wants to report the distance between the starting point and the ending point in meters.

Input: A number N representing the number of pairs of starting and stopping points. N lines of data each containing four values: latitude and longitude of the starting point and the latitude and longitude of the stopping point. All values are in degrees.

Output: For each line of data print “The distance between ” (coordinates of start shown with 5 decimal places) then “ and ” followed by (coordinates of end shown with 5 decimal places) followed by “ is ” followed by a space and the distance travelled in meters rounded to a whole number followed by a space and finally “meters.”. The distance between the points will never exceed Double.MAX_VALUE.

Sample input:

4

```
31.37646 -100.44892 31.36184 -100.43369
30.28017 -97.73871 30.27521 -97.74011
30.28642 -97.73645 30.28771 -97.73333
30.26557 -97.74465 29.76694 -95.37830
```

Sample output:

```
The distance between (31.37646, -100.44892) and (31.36184, -100.43369) is 2176 meters.
The distance between (30.28017, -97.73871) and (30.27521, -97.74011) is 568 meters.
The distance between (30.28642, -97.73645) and (30.28771, -97.73333) is 332 meters.
The distance between (30.26557, -97.74465) and (29.76694, -95.37830) is 234481 meters.
```

11. Reka

Program Name: Reka.java

Input File: reka.dat

Reka and her friend are playing a game. Her friend has a palindromic string of lowercase letters. A palindrome is a string that reads the same forwards and backwards. For example, "racecar" and "moon" are palindromes, while "truck" and "sun" are not palindromes.

Reka has to guess what her friend's string is. She's given L , the length of the string and C clues. Each clue gives the letter in some position of the string. Reka's been guessing for a while, and thinks the game is unfair since there's potentially many possibilities. Write a program to count the number of valid palindromes that are consistent with the given clues.

Input: The first line of input contains an integer T ($1 \leq T \leq 50$), the number of test cases. Each test case begins with two integers L ($1 \leq L \leq 10^{18}$) and C ($0 \leq C \leq \min(L, 200)$), the length of the string and the number of clues. Then follow C lines of clues. Each clue is an index into the string and a lowercase character. Reka and her friend use 1-indexing to refer to the characters in the string. All indexes for a given test case are distinct.

Note that the length of the string will not fit into a 32-bit integer.

Output: For each test case, output the number of different palindromes that are consistent with the clues given. Format your answer with the case number as in the samples. Since the answer can be very very large, output just the last 9 digits of the answer.

Sample Input:

```
3
2 1
2 a
2 2
1 a
2 b
4 0
```

Sample Output:

```
Case #1: 1
Case #2: 0
Case #3: 676
```

Sample Explanation: In the first case, the first character in the string has to be an 'a' to make the result a palindrome, so there is 1 valid answer.

In the second case, all characters are given and the resulting string is not a palindrome. Therefore the answer is 0.

In the third case, there are no clues. Some of the valid palindromic strings of length 4 are "abba" and "zzzz". Funnily enough, the number of ways to make this string a palindrome is a palindrome itself!

12. Timothy

Program Name: Timothy.java

Input File: timothy.dat

Timothy needs to organize a list of whole numbers so that duplicate values are removed, and the list is printed in descending order. In addition, the list needs to indicate how many of each duplicated value have been removed.

Input: A list of whole numbers N where $0 \leq N < 50$. The list will be of unknown length where each value is separated by a space and there will be no more than 20 numbers on each line.

Output: A list of the unique numbers contained in the list sorted in descending order and, for those numbers that were duplicated, the number of duplicates that were removed (not the total number of times the number appeared in the list). Each number should appear on a separate line along with the number of removed duplicates. Separate each pair with a single space. If a value was not duplicated in the list, do not print a second value.

Sample input:

```
26 15 0 17 43 29 0 45 38 43 22 34 9 48 39 43 6 17 49 12
47 13 31 16 47 34 43 46 24 10 31 39 48 7 41 49 23 4 22 6
44 30 43 40 30 23 9 31 35 13 34 1 30 12 42 32 15 7 18 31
37 13 26 10 25 33 22 26 15 34 36 5 41 46 25 11 45 42 5 2
0 10 2 46 21 34 36 37 26 17 49 12 43 41 16 29 26 16 9 41
15 41 33 45 35 3 4 8 24 9 43 30 2 29 7 42 0 25 12 13
```

Sample output: *The output will appear in one column. Two columns are used here for formatting purposes.*

49	2	24	1
48	1	23	1
47	1	22	2
46	2	21	
45	2	18	
44		17	2
43	6	16	2
42	2	15	3
41	4	13	3
40		12	3
39	1	11	
38		10	2
37	1	9	3
36	1	8	
35	1	7	2
34	4	6	1
33	1	5	1
32		4	1
31	3	3	
30	3	2	2
29	2	1	
26	4	0	3
25	2		