JOTHSHANA S M
7376221CS181

PROBLEM 1:
The Character Census Challenge

You find yourself in a whimsical town where the local inhabitants have a unique way of keeping track of the characters in their conversations. They've devised a game called "Character Census Challenge," and you're eager to participate!

The rules are simple: You provide the townsfolk with a string of characters, and they expect you to count the occurrences of each character in the string. However, they want the results to be presented in a quirky way. For each character, you must display the character itself, followed by a dash, and then the count of how many times it appears in the string.

Sample Input: AAAAAAAAJJJJJJDDDDDDDKKKKKKkkklll
Sample Output : A-8, J-6, D-6, K-6, k-3, l-3

Test cases:

1. JJJKKKKSLLLSSSEELLPP
2. AAAAMMMMEEEElllldddddd
3. ZZZZZZZZZZZZZZZZZZZZ
4. 000222kkkkkkkdsDSXSDFCES
5. WIEJFGVNWIEG30945555533
6. MMMMMMMMEEEE999992K2222

Answer for test cases:

1. J-3, K-4, S-4, L-5, E-2, P-2
2. A-4, M-4, E-4, l-4, d-5
3. Z-20
4. 0-3, 2-3, k-7, d-1, s-1, D-2, S-3, X-1, F-1, C-1, E-1
5. W-2, I-2, E-2, J-1, F-1, G-2, V-1, N-1, 3-3, 0-1, 9-1, 4-1, 5-5
6. M-8, E-4, 9-5, 2-5, K-1

PROBLEM 2:
The Palindrome Patrol

In a town known for its love of wordplay, a group of enthusiasts has formed the "Palindrome Patrol." They are on a mission to identify palindromes – words or phrases that read the same backward as forward. The patrol believes that understanding palindromes is the key to unraveling the town's linguistic mysteries.

To join the Palindrome Patrol, you must create a program that checks if a given word or phrase is a palindrome. The patrol is strict, and spaces, punctuation, and letter case should be ignored during the evaluation.

Task:
Write a program that prompts the user to enter a word or phrase and determines if it is a palindrome. Your program should consider only the alphanumeric characters in the evaluation and should be case-insensitive.

For example, if the user enters "A man, a plan, a canal, Panama!", the program should respond with "Yes, it's a palindrome!", if not a palindrome, print "No, it's not a palindrome."

Test cases:

1. MALAYALI
2. CIVIC
3. RABBIT
4. MADAM
5. KAYAK
6. EOSINOPHILS

Answer for test cases:

1. No, it's not a palindrome.
2. Yes, it's a palindrome!
3. No, it's not a palindrome.
4. Yes, it's a palindrome!
5. Yes, it's a palindrome!
6. No, it's not a palindrome.

PROGRAM 3:
The Password Validator

In a town where security is a top priority, the residents have developed a set of rules for creating secure passwords. To be considered secure, a password must adhere to the following rules:
It must have a minimum length of 8 characters.
It must contain at least one uppercase letter.
It must contain at least one lowercase letter.
It must contain at least one digit.
It must contain at least one special character (e.g., !, @, #, $, etc.).
Your challenge is to create a program that validates whether a given password meets these security requirements.
Task:

Write a program that prompts the user to enter a password. Your program should then check if the entered password meets the security requirements and display an appropriate message.

For example:

If the user enters "SecurePwd123!", the program should print "Password is secure."

If the user enters "weak", the program should print "Password is not secure. It does not meet the minimum requirements."

Test cases:

1. AbcdEfgh123@
2. Short1!
3. lowercase1!
4. UPPERCASE1!
5. NoDigits!
6. SpecialCharacter123

Answer for test cases:

1. Password is secure.
2. Password is not secure. It does not meet the minimum length requirement.
3. Password is not secure. It does not contain at least one uppercase letter.
4. Password is not secure. It does not contain at least one lowercase letter.
5. Password is not secure. It does not contain at least one digit.
6. Password is not secure. It does not contain at least one special character.

PROGRAM 4:

The Word Reverser

In a town known for its linguistic puzzles, there is a challenge called the "Word Reverser." The task is to create a program that takes a sentence as input and outputs the sentence with each word reversed while maintaining the order of the words.

For example, if the user enters "Hello World, how are you?" the program should output "olleH dlroW, woh era ?uoy."

Task:

Write a program that prompts the user to enter a sentence. Your program should reverse each word in the sentence while keeping the order of the words, and then display the modified sentence.

Test cases:

1. gnimmargorP si emosewa
2. ehT kciuq nworb xof
3. nohtyP si !nuf

4. olleH dlroW
5. 321 654
6. lufrednow

Answers for test cases:

1. Programming is awesome
2. The quick brown fox
3. Python is fun!
4. Hello World
5. 123 456
6. Wonderful.

PROGRAM 5:
The Unique Elements Finder

In a town where uniqueness is celebrated, there is a challenge known as the "Unique Elements Finder." The task is to create a program that takes a list of numbers as input and outputs a new list containing only the unique elements from the original list.

For example, if the user enters the list 1 2 3 2 4 5 1 the program should output [1, 2, 3, 4, 5].

Task:
Write a program that prompts the user to enter a list of numbers. Your program should then create a new list containing only the unique elements from the original list and display the result.

Test cases:

1. 3 4 4 4 5 5 2 1 7
2. 10 76 32 44 3 6 6 44
3. 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4. 33 3 33 33 33 332 234 4 3 33 3
5. 11 2 2 2 2333 33 3 3 333 3 3 33 33 333333 333333
6. 43 43 43 4323 23 5 533 3221

Answers for test cases:

1. [1, 2, 3, 4, 5, 7]
2. [3, 6, 10, 32, 44, 76]
3. [1, 4]
4. [3, 4, 33, 234, 332]

5. [2, 3, 11, 33, 333, 2333, 333333]
6. [5, 23, 43, 533, 3221, 4323]

PROGRAM 6:
The Fibonacci Explorer

In a town with a fascination for patterns, a new explorer known as the "Fibonacci Navigator" has embarked on a journey to uncover the secrets of the Fibonacci series. The Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1.

Your task is to assist the Fibonacci Navigator by creating a program that generates the first 'n' numbers of the Fibonacci series.

Task:
Write a program that prompts the user to enter a positive integer 'n.' Your program should then generate and display the first 'n' numbers of the Fibonacci series.
For example, if the user enters '5,' the program should output: [0, 1, 1, 2, 3]

Feel free to add informative messages or hints to guide the user through the Fibonacci exploration!

Test cases:
1. 1
2. 11
3. 20
4. 3
5. 7
6. 4

Answers for test cases:
1. [0]
2. [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
3. [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181]
4. [0, 1, 1]
5. [0, 1, 1, 2, 3, 5, 8]
6. [0, 1, 1, 2]

PROGRAM 7:
The Time Puzzle Solver

In a small town with a penchant for puzzles, there's a challenge known as the "Time Puzzle Solver." The task is to determine whether the bits of paper drawn from three bowls—Hour Bowl, Minute Bowl, and Seconds Bowl—can be arranged to form a valid time on the clock.

Each bowl contains bits of paper numbered from 1 to 100. The goal is to draw one bit from each bowl and arrange them to create a valid time on the clock.

Task:
Write a program that simulates the Time Puzzle Solver. Prompt the user to enter three numbers, each representing a bit drawn from the Hour, Minute, and Seconds bowls. Your program should then check whether these bits can be arranged to form a valid time.
Your program should output "Valid" if the bits can form a valid time and "Not valid" otherwise.
For example, if the user enters 5:30:45 the program should output "Valid".

Test cases:

1. 24:32:45
2. 23:60:21
3. 12:23:45
4. 23:59:59
5. 6:4:3
6. 22:60:50

Answers for test cases:

1. Not valid
2. Not valid
3. Valid
4. Valid
5. Valid
6. Not valid

PROGRAM 8:
The Alternate Digit Swapper

In a town where numerical patterns are a source of curiosity, a challenge known as the "Alternate Digit Swapper" has captured the attention of the residents. The task is to create a program that swaps adjacent digits in a number alternately, creating a new and intriguing sequence.
Your task is to design a program that takes a number as input and performs the following alternate digit swapping operation:

Starting from the leftmost digit, swap it with the next digit.
Move two positions to the right and swap the next pair of digits.

Continue this process until the end of the number is reached.

Task:
Write a C program that simulates the Alternate Digit Swapper. Prompt the user to enter a number, and your program should then perform the alternate digit swapping operation to create a new sequence.
For example, if the user enters '12345' the program should output '21435' after swapping adjacent digits alternately.

Test cases:

1. 123456789
2. 22221
3. 9898766
4. 1234432332233322
5. 54453423232376544
6. 0897654325

Answers for test cases:

1. 214365879
2. 22221
3. 8989676
4. 2143343223323322
5. 45544332323267454
6. 807956789

PROGRAM 9:
The Sentence Transformer

In a quaint town known for its unique writing styles, there's a challenge called the "Sentence Transformer." The task is to transform ordinary sentences into a special form called Camel Case. Camel Case is a style of writing where spaces are removed, and each word begins with a capital letter.

Task:
Write a program that takes a sentence as input and transforms it into Camel Case. The program should remove spaces from the sentence and rewrite it in the Camel Case format.

For example, if the user enters "here comes-the.garden," the program should output "HereComesTheGarden" after the Camel Case transformation.

Test cases:

1. Those are very tall
2. Camel Case is Fun
3. 123 numbers are included
4. one

5. hello@@@ - world
6. hi-guys

Answers for test cases:

1. ThoseAreVeryTall
2. CamelCaseIsFun
3. 123NumbersAreIncluded
4. One
5. HelloWorld
6. HiGuys

PROGRAM 10:
The Prime Quest

In a secret realm known as HackerClub, a brilliant hacker named Joy faces a fascinating challenge—the "Prime Quest." Joy's mission is to find the nearest prime number for a given integer N. The task becomes even more intriguing as multiple answers might exist, and Joy must output the smallest among them. As Joy dives into this quest, he encounters T different scenarios.

Task:

Write a program that aids Joy in the Prime Quest. The program should take an integer T, representing the number of test cases. For each test case, it should find and output the nearest prime number to N, with a preference for the smallest prime when multiple solutions are possible.

Sample input:

3
51
12
65

Sample output:

53
11
67

Test cases:

1. 5
   34
   87
   45
   83
   39

2. 2
   98
   54

3. 3
   39
   94
   100
4. 2
   75
   28
5. 5
   95
   35
   72
   36
   69
6. 2
   49
   85

Answers for test cases:

1. 31
   89
   43
   79
   37
2. 97
   53
3. 37
   97
   101
4. 73
   29
5. 97
   37
   71
   37
   67
6. 47
   83

PROGRAM 11: Done

Prime Number Discovery

In a community dedicated to mathematical exploration, an initiative known as the "Prime Number Discovery" has been launched. Enthusiasts are encouraged to embark on a journey to uncover and understand prime numbers within a specified range. As part of this initiative, a program is needed to systematically list all prime numbers within the given boundaries.

Task:
Write a program that facilitates the Prime Number Discovery. Prompt the user to enter a range defined by two integers: a starting number and an ending number. The program should then meticulously calculate and present all prime numbers within the specified range.

Test cases:

1. 0 15
2. 15 40
3. 20 40
4. 40 60
5. 90 120
6. 30 50

Answers for test cases:

1. 2 3 5 7 11 13
2. 17 19 23 29 31 37
3. 23 29 31 37
4. 41 43 47 53 59
5. 97 101 103 107 109 113
6. 31 37 41 43 47

PROGRAM 12:
The Harmonious String Symphony

In a picturesque town where every word is a melody, a challenge known as the "Harmonious String Symphony" has arisen. The task is to create a program that brings harmony to the strings by removing adjacent duplicate characters. Residents aspire for a flowing rhythm in their sentences, desiring to eliminate only consecutive same characters, allowing for a harmonious string composition.

Task:

Write a program that plays a pivotal role in the Harmonious String Symphony. Prompt the user to enter a string, and the program should skillfully remove adjacent duplicate characters, maintaining the musical flow of the text.

Test cases:

1. aaabbcdddeffgtaaaaaaaaa - **sample**
2. aaaaaaaaaaaaa
3. ccccccccooommmpppuuuttteerr
4. ccccccchhhhhaaaarggggggggerrrr
5. lllllllllaaaaaaaaalllllllllll
6. fffffffinnnnnnggggggggggggerrrrrrrrrrrrrrrrr
7. sssssssssssyyyyyssssttttttttttttttttttttttttttttttttteeemmmmmm

Answers for test cases:

1. abcdefgta - **sample**
2. a
3. computer
4. Charger
5. lal
6. Finger
7. system

PROGRAM 13:
The Word Reversal Quest done

In a town that cherishes linguistic mysteries, there's a challenge called the "Word Reversal Quest." The task is to create a program that assists in unraveling the enigma of word reversal. Residents are intrigued by the notion of exploring words in reverse order within a sentence.

Task:
Write a program that plays a crucial role in the Word Reversal Quest. Prompt the user to enter a sentence, and the program should skillfully reverse the order of words within that sentence, creating a captivating linguistic journey.

Sample input: I have a pen with me
Sample output: me with pen a have I

Test cases:

1. What is your name
2. Handle it carefully

3. Learn books and gain knowledge
4. Get well soon
5. I am an ordinary person
6. Bring me a cup of tea

Answers for test cases:

1. name your is What
2. carefully it Handle
3. knowledge gain and books Learn
4. soon well Get
5. person ordinary an am I
6. tea of cup a me Bring

PROGRAM 14:
The Vowel-Less Word Mastery

In a cozy living room where the bond between a father and his daughter flourishes, a delightful game unfolds — the "Vowel-Less Word Mastery." The rules are simple yet engaging: the father writes a word on a piece of paper, and the daughter's challenge is to skillfully eliminate the vowels and then pronounce the transformed word. As they embark on this linguistic journey, they seek assistance to ensure the pronunciation is just as charming without the presence of vowels.

Task:
Write a program that aids the daughter in the Vowel-Less Word Mastery game. The program should prompt the user (the daughter) to enter a word, eliminate the vowels, and present the transformed word for pronunciation.

Test cases:

1. apple
2. chocolate
3. banana
4. table
5. library
6. window

Answers for test cases:

1. ppl
2. chclt
3. bnn

4. tbl
5. lbrry
6. wndw

PROGRAM 15:
The Binary Alternator

In a digital village known for its fascination with binary rhythms, there's a captivating challenge called the "Binary Alternator." The task is to create a program that orchestrates an alternating pattern of 1's and 0's, resembling the beats of a binary drum. Villagers are eager to explore this rhythmic landscape and create harmonious patterns.

Task:
Write a program that takes center stage in the Binary Alternator challenge. Prompt the user to enter a number, and the program should craft an alternating pattern of 1's and 0's, forming a rhythmic sequence that delights the residents.

Test cases:
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7 **-sample**

Answers for test cases:
1. 1
2. 1 1
   0 0
3. 1 1 1
   0 0 0
   1 1 1
4. 1 1 1 1
   0 0 0 0
   1 1 1 1
   0 0 0 0

5. 1 1 1 1 1

0 0 0 0 0
1 1 1 1 1
0 0 0 0 0
1 1 1 1 1

6. 1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 1 1 1
0 0 0 0 0 0

7. 1 1 1 1 1 1 1
0 0 0 0 0 0 0
1 1 1 1 1 1 1
0 0 0 0 0 0 0 **- sample**
1 1 1 1 1 1 1
0 0 0 0 0 0 0
1 1 1 1 1 1 1

PROGRAM 16:
The Secret Code Generator

In a mysterious village where codes are the language of choice, a challenge known as the "Secret Code Generator" has surfaced. The task is to create a program that generates secret codes based on a given key. Villagers are curious to encode their messages using this unique system.

Task:
Write a program that takes center stage in the Secret Code Generator challenge. Prompt the user to enter a message and a numerical key, and the program should generate a secret code by shifting each letter in the message according to

the key. Residents can use this secret code to share hidden messages.(number remains unchanged)

Test cases:

1. Hello
   3
2. Hello World
   5
3. Python
   3
4. 12345
   2
5. Alphabet
   1
6. Test Case
   6

Answers for test cases:

1. Khoor
2. Mjqqt Btwqi
3. Sbwkrq
4. 12345
5. Bmqibcfu
6. Zkyz Igyk

PROGRAM 17:
The Numeric Palindrome Explorer

In a town fascinated by numerical wonders, a challenge known as the "Numeric Palindrome Explorer" has emerged. The task is to create a program that explores the world of numeric palindromes. Residents are eager to identify palindromic numbers within a given range and marvel at their symmetrical beauty.

Task:
Write a program that takes center stage in the Numeric Palindrome Explorer challenge. Prompt the user to enter a range of numbers, and the program should identify and display all the palindromic numbers within that range.

Test cases:

1. 100
   150
2. 100
   200
3. 1000

2000
4. 3000
3500
5. 10000
10500
6. 1
50

Answers for test cases:

1. [101, 111, 121, 131, 141]
2. [101, 111, 121, 131, 141, 151, 161, 171, 181, 191]
3. [1001, 1111, 1221, 1331, 1441, 1551, 1661, 1771, 1881, 1991]
4. [3003, 3113, 3223, 3333, 3443]
5. [10001, 10101, 10201, 10301, 10401]
6. [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44]

PROGRAM 18:
The Grade Analyzer

In a town dedicated to education, there's a challenge known as the "Grade Analyzer." The task is to create a program that helps students and teachers analyze grades. Residents are interested in understanding the distribution of grades and identifying high-performing students.

Task:

Write a program that takes center stage in the Grade Analyzer challenge. Prompt the user to enter a list of student grades, and the program should analyze and categorize the grades into different bands (A, B, C, D, F). Additionally, it should identify the highest and lowest grades.

Test cases:

1. 85, 92, 78, 94, 60, 72, 89, 95, 88, 75
2. 30, 40, 53, 51, 60, 65, 68, 62, 75, 72, 89, 84, 90, 96
3. 34, 60, 64, 68, 73, 79, 43, 28, 74, 80, 86, 83, 91, 95, 97
4. 70, 78, 74, 85, 82, 67, 71, 86, 82, 90, 97
5. 94, 96, 80, 74, 60, 32, 50, 69
6. 90, 80, 50, 30, 20, 50, 60, 75, 83, 94, 98, 68, 52

Answers for test cases:

1. Grade A: 3 students
   Grade B: 3 students
   Grade C: 3 students
   Grade D: 1 students
   Grade F: 0 students
   Highest Grade: 95
   Lowest Grade: 60

2. Grade A: 2 students
   Grade B: 2 students
   Grade C: 2 students
   Grade D: 4 students
   Grade F: 4 students
   Highest Grade: 96
   Lowest Grade: 30

3. Grade A: 3 students
   Grade B: 3 students
   Grade C: 3 students
   Grade D: 3 students
   Grade F: 3 students
   Highest Grade: 97
   Lowest Grade: 28

4. Grade A: 2 students
   Grade B: 4 students
   Grade C: 4 students
   Grade D: 1 students
   Grade F: 0 students
   Highest Grade: 97
   Lowest Grade: 67

5. Grade A: 2 students
   Grade B: 1 students
   Grade C: 1 students
   Grade D: 2 students
   Grade F: 2 students
   Highest Grade: 96
   Lowest Grade: 32

6. Grade A: 3 students
   Grade B: 2 students
   Grade C: 1 students
   Grade D: 2 students
   Grade F: 5 students

Highest Grade: 98
Lowest Grade: 20

PROGRAM 19:
The Anagram Checker

In a town where words create intricate connections, there's a challenge known as the "Anagram Checker." The task is to create a program that checks if two words are anagrams of each other. Residents are interested in discovering the delightful symphony of letters that form anagrams.

Task:
Write a program that takes center stage in the Anagram Checker challenge. Prompt the user to enter two words, and the program should determine whether the words are anagrams of each other or not.

Test cases:

1. listen
   silent
2. brush
   shrub
3. elegant
   gentle
4. customers
   store scum
5. redo it
   editor
6. problems
   robbers

Answer for test cases:

1. Anagrams!
2. Anagrams!
3. Not Anagrams!
4. Anagrams!
5. Anagrams!
6. Not Anagrams!

PROBLEM 20:
The Password Strength Checker

In a town where cybersecurity is a top priority, a challenge known as the "Password Strength Checker" has emerged. The task is to create a program that helps residents assess the strength of their passwords. Security-conscious

individuals want a tool that provides feedback on the robustness of their chosen passwords.

Task:
Write a program that takes center stage in the Password Strength Checker challenge. Prompt the user to enter a password, and the program should analyze and categorize the strength of the password based on various criteria such as length, use of uppercase letters, lowercase letters, numbers, and special characters.

Test cases:

1. P@ssw0rd123!
2. Laptop123
3. C0mpl3xP@ss!
4. Saf3P@ssw0rd!
5. password@10000
6. Laptop@
7. laptop@
8. LARVA@123

Answers for test cases:

1. Strong
2. Weak
3. Strong
4. Strong
5. Weak
6. Weak
7. Weak
8. Weak

PROGRAM 21:
The Number Triangle

In a town with a fascination for patterns, a challenge known as the "Number Triangle" has emerged. The task is to create a program that generates a numerical triangle pattern. Residents are intrigued by the mathematical beauty of triangular patterns and wish to explore them further.

Task:
Write a program that takes center stage in the Number Triangle challenge. Prompt the user to enter the height of the triangle (number of rows), and the program should generate a numerical triangle pattern as follows:

Sample input: 5

Sample output:

1
22
333
4444
55555

Test cases:

1. 3
2. 2
3. 5
4. 7
5. 6
6. 4

Answers for test cases:

1. 1
   22
   333
2. 1
   22

3. 1
   22
   333
   4444
   55555
4. 1
   22
   333
   4444
   55555
   666666
   7777777
5. 1
   22
   333
   4444
   55555
   666666
6. 1
   22
   333

4444

PROGRAM 22:
The Word Frequency Counter

In a town filled with wordsmiths, a challenge known as the "Word Frequency Counter" has emerged. The task is to create a program that analyzes a given text and counts the frequency of each word. Residents are interested in understanding the prominence of words in their writings.

Task:
Write a program that takes center stage in the Word Frequency Counter challenge. Prompt the user to enter a text, and the program should analyze the text to count and display the frequency of each word.

Test cases:

1. The quick brown fox jumps over the lazy dog THE QUICK BROWN DOG
2. Hey! How's it going? Hey, it's good. It's going well
3. the quick brown fox jumps over the lazy dog the quick brown dog

Answer for test cases:

1. the: 3
   quick: 2
   brown: 2
   fox: 1
   jumps: 1
   over: 1
   lazy: 1
   dog: 2
2. hey: 2
   how's: 1
   it: 2

going: 2
hey: 1
it's: 2
good: 1
well: 1

3. the: 3
quick: 2
brown: 2
fox: 1
jumps: 1
over: 1
lazy: 1
dog: 2

PROGRAM 23:
The Unique Element Finder

In a town known for its unique artifacts, a challenge known as the "Unique Element Finder" has emerged. The task is to create a program that helps residents identify the unique element in a list of integers. The townspeople are

fascinated by rare finds and want a tool to uncover the unique element in a collection.

Task:
Write a program that takes center stage in the Unique Element Finder challenge. Prompt the user to enter a list of integers, and the program should identify and display the count of numbers which was again repeated in the array/list. The unique element is the one that occurs only once, while the others may have duplicates.

Test cases:

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

Answers for test cases:

1. 6
2. 0
3. 4
4. 4
5. 3
6. 2

PROGRAM 24:
The Art Gallery Inventory

In a town with a vibrant art community, a challenge known as the "Art Gallery Inventory" has emerged. The town's art curator is fascinated by the uniqueness of each artwork and wants to know how many distinct pieces are currently in the gallery. Residents are encouraged to participate in this artful endeavor.

Task:
Write a program that takes center stage in the Art Gallery Inventory challenge. The program prompts users to input a list of artwork identifiers, where each identifier represents a unique piece in the gallery. The curator wants to determine the total count of distinct artworks.

Test cases:

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

Answers for test cases:

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

PROGRAM 25:
The Shopping Cart Calculator

In a bustling town filled with avid shoppers, a challenge known as the "Shopping Cart Calculator" has emerged. Residents want a tool that helps them keep track of their shopping expenses and apply discounts. The task is to create a program that calculates the total cost of items in a shopping cart, considering any applicable discounts.

Task:
Write a program that takes center stage in the Shopping Cart Calculator challenge. The program prompts users to input the names and prices of items in their shopping cart. It should then apply discounts based on the total cost of the items. For example, if the total cost is above $100, apply a 10% discount.The loop must go for as many as input that user gives. If the products that the user bought is over, then allow the user to type “done” to complete the input phase.

Eg:

Sample input 1:

A
20
B
30
C
40

done
Sample output 1:
A:20.00
B:30.00
C:40.00

Total:90.00
Final:90.00

Sample input 2:
A
50
B
90
done
Sample output 2:
a:50.00
b:90.00
Tota:140.00
Discounts:14.00 (10%)
Final:126.00

Test cases:

1. A
   20
   B
   34
   C
   20
   done
2. A
   20
   B
   100
   C
   300
   done
3. A
   40
   B
   200
   C
   400

D
300
done

4. a
101
done

Answers for test cases:

1. A:20.00
B:34.00
C:20.00
Total:74.00
Fina:74.00
2. A:20.00
B:100.00
C:300.00
Total:420.00
Discounts:42.00 (10%)
Final:378.00
3. A:40.00
B:200.00
C:400.00
D:300.00
Total:940.00
Discounts:94.00 (10%)
Final:846.00
4. a:101.00
Total:101.00
Discounts:10.10 (10%)
Final:90.90

PROGRAM 26:
The Secret Code Generator

In a town fascinated by mysteries and secrets, a challenge known as the "Secret Code Generator" has emerged. The task is to create a program that allows users to encode and decode messages using a secret code. The program should use a

simple algorithm to transform messages into coded versions and decode them back.

Task:

Write a program that takes center stage in the Secret Code Generator challenge. The program should prompt users to input a message and choose whether they want to encode or decode it. It should then apply the secret code algorithm and display the result.encode-1, decode-2. Encode changes each letter to it's corresponding ascii value. And decode works from ascii value to letters.

Eg: **input:** mind blowing

1

output:109-105-110-100-32-98-108-111-119-105-110-103

Test cases:

1. Personalized skill
   1
2. Programming area
   1
3. 99-117-112-32-111-102-32-99-111-102-102-101-101
   2
4. 114-97-105-110-32-97-110-100-32-114-97-105-110-98-111-119
   2
5. Looks good
   1
6. Happy morning!
   1

Answers for test cases:

1. 80-101-114-115-111-110-97-108-105-122-101-100-115-107-105-108-108
2. 80-114-111-103-114-97-109-109-105-110-103-32-97-114-101-97
3. cup of coffee
4. rain and rainbow
5. 76-111-111-107-115-32-103-111-111-100
6. 72-97-112-112-121-32-109-111-114-110-105-110-103-33

PROGRAM 27

The Treasure Map Challenge

In a distant land, there's a legendary treasure hidden in a vast desert, and adventurers are drawn to the challenge of finding it. The only clue to the treasure's location is a mysterious map encoded with symbols.

The map is represented as a string, where each symbol corresponds to a specific direction:

"N" for North

"S" for South

"E" for East

"W" for West

Task:

Help the adventurers determine the final coordinates of the treasure based on the movements indicated by the map.

Example:

Sample input: NNESWW

Sample output: (-1, 1)

Test cases:

1. NENWSE
2. SSSNEEEWWW
3. NESNWSSWW
4. NESWWNSSWWE
5. NNNNNNSSSEEEWWW
6. NNNSSSEEEWW

Answers for test cases:

1. (1, 1)
2. (0, -2)
3. (-2, -1)
4. (-2, -1)
5. (0, 3)
6. (1, 0)

ANSWERS:

1.
```
def count_characters(input_string):
    counts = {}
    for char in input_string:
        if char in counts:
            counts[char] += 1
        else:
            counts[char] = 1

    output = []
    for char, count in counts.items():
        output.append(f"{char}-{count}")

    return ", ".join(output)

# Get user input
user_input = input("Enter a string: ")
result = count_characters(user_input)
print(result)
```

2.
```
def is_palindrome(s):
    s = ''.join(char.lower() for char in s if char.isalnum())
    return s == s[::-1]
user_input = input("Enter a word or phrase: ")
if is_palindrome(user_input):
    print("Yes, it's a palindrome!")
else:
    print("No, it's not a palindrome.")
```

3.
```
import re
```

```
def is_secure_password(password):
    # Check minimum length
    if len(password) < 8:
        return False, "Password is not secure. It does not meet the minimum length requirement."

    # Check for at least one uppercase letter
    if not any(char.isupper() for char in password):
        return False, "Password is not secure. It does not contain at least one uppercase letter."

    # Check for at least one lowercase letter
    if not any(char.islower() for char in password):
        return False, "Password is not secure. It does not contain at least one lowercase letter."

    # Check for at least one digit
    if not any(char.isdigit() for char in password):
        return False, "Password is not secure. It does not contain at least one digit."

    # Check for at least one special character
    if not re.search("[!@#$%^&*()-_+=]", password):
        return False, "Password is not secure. It does not contain at least one special character."

    return True, "Password is secure."

# Get user input for the password
user_password = input("Enter a password: ")

# Validate the password and display the result
is_secure, message = is_secure_password(user_password)
print(message)
```

4. def reverse_words(sentence):
    words = sentence.split()
    reversed_words = [word[::-1] for word in words]
    reversed_sentence = ' '.join(reversed_words)
    return reversed_sentence

# Get user input for the sentence
user_sentence = input("Enter a sentence: ")

# Reverse the words in the sentence and display the result
reversed_result = reverse_words(user_sentence)
print("Reversed Sentence:", reversed_result)

5. def find_unique_elements(input_list):
    unique_elements = list(set(input_list))
    unique_elements.sort()
    return unique_elements

# Get user input for the list of numbers
user_input = input("Enter a list of numbers separated by spaces: ")
user_list = [int(num) for num in user_input.split()]

# Find and display the unique elements in the list
unique_elements_result = find_unique_elements(user_list)
print("Unique Elements:", unique_elements_result)

6. def generate_fibonacci(n):
    fibonacci_series = []

```
    a, b = 0, 1

    for _ in range(n):
        fibonacci_series.append(a)
        a, b = b, a + b

    return fibonacci_series

# Get user input for the number of Fibonacci numbers to generate
n = int(input("Enter the number of Fibonacci numbers to generate: "))

# Generate and display the Fibonacci series
fibonacci_result = generate_fibonacci(n)
print("Fibonacci Series:", fibonacci_result)
```

7.
```
#include <stdio.h>

int isValidTime(int hour, int minute, int second) {
    return (hour >= 0 && hour <= 23) &&
           (minute >= 0 && minute <= 59) &&
           (second >= 0 && second <= 59);
}

int main() {
    int hour, minute, second;

    // Get user input for bits from each bowl
    scanf("%d:%d:%d", &hour, &minute, &second);

    // Check if the bits can form a valid time
    if (isValidTime(hour, minute, second)) {
```

```
        printf("Valid\n");
    } else {
        printf("Not valid\n");
    }

    return 0;
}
```

8.
```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<stdlib.h>

int reverse(int num){
    int rev = 0;
    while(num!=0){
        rev=rev*10+(num%10);
        num=num/10;
    }
    return rev;
}

int main(){
    int i=1,n,c=0,no=0,t,s;
    scanf("%d",&n);
    t = n;
    while(t!=0){
        c++;
        t=t/10;
    }
    s=reverse(n);
```

```
    while(i<=(c/2)){
        no=no*100+(s%100);
        s=s/100;
        i++;
    }
    if(c%2!=0){
        no=no*10+(n%10);
    }
    printf("%d", no);
    return 0;
}
```

9.
```
import re
def camel_case(sentence):
    sentence = re.sub(r"[^\w\s]"," ", sentence).strip()
    sentence = sentence.title()
    sentence = sentence.replace(" ", "")
    return sentence

sentence = input()
camel_case_string = camel_case(sentence)
print(camel_case_string)
```

10.
```
import math
def is_prime(num):
    if num < 2:
        return False
    for i in range(2,int(math.sqrt(num))+1):
        if num%i == 0:
            return False
    return True
```

```
def next_prime(num):
    while True:
        num+=1
        if is_prime(num):
            return num
def prev_prime(num):
    while True:
        num-=1
        if is_prime(num):
            return num
T = int(input())
for _ in range(T):
    N = int (input())
    left = prev_prime(N)
    right = next_prime(N)
    if(N - left)<=(right - N):
        print(left)
    else:
        print(right)
```

11.

12.
```
#include<stdio.h>
#include<string.h>

void removee(char inp_str[]){
    int length = strlen(inp_str);
    if (length <= 1){
        return;
    }
```

```
    int j = 0;
    for(int i = 1;i < length; i++){
        if(inp_str[i] != inp_str[j]){
            j++;
            inp_str[j] = inp_str[i];
        }
    }
    inp_str[j + 1] = '\0';
}

int main(){
    char input_string[100];
    fgets(input_string, sizeof(input_string), stdin);
    input_string[strcspn(input_string, "\n")] = '\0';

    removee(input_string);
    printf("%s\n", input_string);
    return 0;
}
```

13.
```
def reverse_words(sentence):
    words = sentence.split()
    reversed_sentence = ' '.join(reversed(words))
    return reversed_sentence

# Get user input for the sentence
user_sentence = input("Enter the sentence: ")

# Reverse the order of words and display the result
reversed_result = reverse_words(user_sentence)
print("Reversed sentence:", reversed_result)
```

14.
```
def remove_vowels(word):
    vowels = "AEIOUaeiou"
    vowel_less_word = ''.join(char for char in word if char not in vowels)
    return vowel_less_word

# Get user input for the word
user_word = input("Enter the word: ")

# Remove vowels and display the result
vowel_less_result = remove_vowels(user_word)
print("Pronounce without vowels:", vowel_less_result)
```

15.
```
def binary_alternator(rows):
    for i in range(rows):
        pattern = ' '.join(['1' if i % 2 == 0 else '0' for _ in range(rows)])
        print(pattern)

# Get user input for the number of rows
user_rows = int(input("Enter the number of rows: "))

# Generate and display the Binary Alternator pattern
binary_alternator(user_rows)
```

16.
```
def generate_secret_code(message, key):
    secret_code = ''
    for char in message:
        if char.isalpha():
            shift = ord('A') if char.isupper() else ord('a')
            coded_char = chr((ord(char) - shift + key) % 26 + shift)
            secret_code += coded_char
        else:
```

```
            secret_code += char
    return secret_code

# Get user input for the message and key
user_message = input("Enter the message: ")
user_key = int(input("Enter the key: "))

# Generate and display the Secret Code
secret_code_result = generate_secret_code(user_message, user_key)
print("Secret Code:", secret_code_result)
```

17.
```
def is_palindrome(number):
    # Convert the number to a string for easy comparison
    str_number = str(number)
    return str_number == str_number[::-1]

def palindrome_explorer(start, end):
    palindromic_numbers = [num for num in range(start, end + 1) if is_palindrome(num)]
    return palindromic_numbers

# Get user input for the range
user_start = int(input("Enter the starting number: "))
user_end = int(input("Enter the ending number: "))

# Explore and display the Numeric Palindromes
palindromic_results = palindrome_explorer(user_start, user_end)
print(f"Palindromic Numbers in the range {user_start} to {user_end}:", palindromic_results)
```

18.
```
def analyze_grades(grades):
    grade_counts = {'A': 0, 'B': 0, 'C': 0, 'D': 0, 'F': 0}
```

```
    for grade in grades:
        if grade >= 90:
            grade_counts['A'] += 1
        elif grade >= 80:
            grade_counts['B'] += 1
        elif grade >= 70:
            grade_counts['C'] += 1
        elif grade >= 60:
            grade_counts['D'] += 1
        else:
            grade_counts['F'] += 1

    highest_grade = max(grades)
    lowest_grade = min(grades)

    return grade_counts, highest_grade, lowest_grade

# Get user input for the list of student grades
user_grades = list(map(int, input("Enter the list of student grades (comma-separated): ").split(',')))

# Analyze and display the Grade Analysis
grade_analysis, highest_grade, lowest_grade = analyze_grades(user_grades)

print("Grade Analysis:")
for grade, count in grade_analysis.items():
    print(f"- Grade {grade}: {count} students")

print("\nHighest Grade:", highest_grade)
print("Lowest Grade:", lowest_grade)
```

19. def are_anagrams(word1, word2):

```
    # Remove spaces and convert to lowercase for case-insensitive comparison
    word1_clean = ''.join(word1.split()).lower()
    word2_clean = ''.join(word2.split()).lower()

    # Check if the sorted characters of both words match
    return sorted(word1_clean) == sorted(word2_clean)

# Get user input for the two words
user_word1 = input("Enter the first word: ")
user_word2 = input("Enter the second word: ")

# Check and display the result
if are_anagrams(user_word1, user_word2):
    print(f"Anagrams!")
else:
    print(f"Not Anagrams!")

20. def check_password_strength(password):
    # Define criteria for password strength
    length_criteria = 8
    uppercase_criteria = lowercase_criteria = number_criteria = special_char_criteria = False

    # Check length criterion
    if len(password) >= length_criteria:
        length_criteria = True

    # Check other criteria
    for char in password:
        if char.isupper():
            uppercase_criteria = True
        elif char.islower():
```

```
            lowercase_criteria = True
        elif char.isdigit():
            number_criteria = True
        elif char in "!@#$%^&*()-_+=<>?":
            special_char_criteria = True

    # Assess overall strength
    if all([length_criteria, uppercase_criteria, lowercase_criteria, number_criteria,
special_char_criteria]):
        return "Password Strength: Strong"
    else:
        return "Password Strength: Weak"

# Get user input for the password
user_password = input("Enter the password: ")

# Check and display the result
result = check_password_strength(user_password)
print(result)
```

21.
```
def generate_number_triangle(height):
    for i in range(1, height + 1):
        print(str(i) * i)

# Get user input for the height of the triangle
user_height = int(input("Enter the height of the triangle: "))

# Generate and display the number triangle
generate_number_triangle(user_height)
```

22.
```
def word_frequency_counter(text):
```

```
    # Split the text into words
    words = text.split()

    # Create a dictionary to store word frequencies
    word_frequency = {}

    # Count the frequency of each word
    for word in words:
        # Remove punctuation and convert to lowercase for accurate counting
        clean_word = word.strip('.,!?"\'').lower()

        # Update the word frequency dictionary
        word_frequency[clean_word] = word_frequency.get(clean_word, 0) + 1

    # Display the word frequency
    print("Word Frequency:")
    for word, frequency in word_frequency.items():
        print(f"{word}: {frequency}")

# Get user input for the text
user_text = input("Enter the text: ")

# Analyze and display the word frequency
word_frequency_counter(user_text)
```

23.
```
def find_unique_elements(numbers):
    # Create a dictionary to store the frequency of each element
    element_frequency = {}

    # Count the frequency of each element
    for num in numbers:
```

```
        element_frequency[num] = element_frequency.get(num, 0) + 1

    # Find and display the count of unique elements
    unique_elements_count = sum(1 for frequency in element_frequency.values() if frequency == 1)

    return unique_elements_count

# Get user input for the list of integers
user_input = input("Enter a list of integers (separated by spaces): ")
numbers = list(map(int, user_input.split()))

# Find and display the count of unique elements
unique_elements_count = find_unique_elements(numbers)
print(f"Count of Unique Elements: {unique_elements_count}")
```

24.
```
def count_unique_elements(numbers):
    # Use a set to keep track of unique elements
    unique_elements = set(numbers)

    # Count the number of unique elements
    count = len(unique_elements)

    return count

# Get user input for the list of integers
user_input = input("Enter a list of integers (separated by spaces): ")
numbers = list(map(int, user_input.split()))

# Count and display the number of unique elements
unique_count = count_unique_elements(numbers)
print(f"Number of Unique Elements: {unique_count}")
```

25.
```
def calculate_discount(total_cost):
    # Apply a 10% discount if the total cost is above $100
    if total_cost > 100:
        discount_percentage = 10
        discount_amount = (discount_percentage / 100) * total_cost
        return discount_amount, discount_percentage
    else:
        return 0, 0

def main():
    shopping_cart = {}

    # Collect items and prices in the shopping cart
    while True:
        item_name = input("Enter item name (type 'done' to finish): ")
        if item_name.lower() == 'done':
            break
        item_price = float(input("Enter item price: "))
        shopping_cart[item_name] = item_price

    # Display itemized list of purchases
    print("\nShopping Cart:")
    total_cost = 0
    for item, price in shopping_cart.items():
        print(f"- {item}: ${price:.2f}")
        total_cost += price

    # Calculate discounts and final cost
    discount_amount, discount_percentage = calculate_discount(total_cost)
    final_cost = total_cost - discount_amount
```

```
    # Display total cost, discounts, and final cost
    print(f"\nTotal Cost Before Discounts: ${total_cost:.2f}")
    if discount_amount > 0:
        print(f"Applied Discounts: ${discount_amount:.2f} ({discount_percentage}%)")
    print(f"Final Cost After Discounts: ${final_cost:.2f}")

if __name__ == "__main__":
    main()

26. def encode_message(message):
    encoded = '-'.join(str(ord(char)) for char in message)
    return encoded

def decode_message(encoded_message):
    try:
        decoded = ''.join(chr(int(code)) for code in encoded_message.split('-'))
        return decoded
    except ValueError:
        return "Invalid encoded message. Please check and try again."

def main():
    message = input("Enter a message: ")

    choice = input("Choose an option:\n1. Encode\n2. Decode\nEnter your choice: ")

    if choice == '1':
        encoded_message = encode_message(message)
        print(f"Encoded Message: {encoded_message}")
    elif choice == '2':
        decoded_message = decode_message(message)
```

```
        print(f"Decoded Message: {decoded_message}")
    else:
        print("Invalid choice. Please choose 1 for encoding or 2 for decoding.")

if __name__ == "__main__":
    main()
```

27.
```
def find_treasure_coordinates(map_string):
    x, y = 0, 0  # Initial coordinates

    # Iterate through each symbol in the map string
    for move in map_string:
        if move == "N":
            y += 1
        elif move == "S":
            y -= 1
        elif move == "E":
            x += 1
        elif move == "W":
            x -= 1

    return x, y

# Get input from the user
map_string = input("Enter the treasure map string: ")

# Call the function to find the coordinates
final_coordinates = find_treasure_coordinates(map_string)

# Display the result
print(f"Final Coordinates: {final_coordinates}")
```

SAMPLE QUESTIONS

Doubling the Rent

01.Problem

Ram decided to move into Jam's apartment.
Jam was initially paying a rent of X rupees. Since Ram is moving in, the owner decided to double the rent.
Find the final rent Jam needs to pay.

Input Format

The input consists of a single integer X, denoting the rent Jam was initially paying.

Output Format

Output on a new line, the final rent Jam needs to pay.

Test case 01:

```
Input
2
Output
4
```

Test case 02:

```
Input
3
Output
6
```

Test case 03:

```
Input
10
Output
20
```

Test case 04:

```
Input
1
Output
2
```

Test case 05:

```
Input
5
Output
10
```

Test case 06:

```
Input
7
Output
14
```

02.Problem

Maximum Water Container

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the i^{th} line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.

Notice that you may not slant the container.

Test case 01:

Input
height = [1, 2, 3, 4, 5]
Output
6

Test case 02:

Input
height = [10000, 10000]
Output
10000

Test case 03:

Input
height = [1,5,2,4,3]
Output
9

Test case 04:

Input
height = [3,3,3,3,3]
Output
12

Test case 05:

Input
height = [1,1]
Output
1

Test case 06:

Input
height = [3,1,5,2,7]
Output
12

Test case 07:

Input
height = [1,8,6,2,5,4,8,3,7]
Output
49

Test case 08:

Input
height = [0,0]
Output
0

Test case 09:

Input
height = [0,10000,0,10000]
Output
20000

03.Problem

Stock Trading Profit Maximization

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Test case 01:

Input
prices =[7,1,5,3,6,4]
Output
5

Test case 02:

Input

prices =[7,6,4,3,1]

Output

0

Test case 03:

Input

prices =[1,2,3,4,5]

Output

4

Test case 04:

Input

prices =[5,4,3,2,1]

Output

0

Test case 05:

Input

prices =[3,3,3,3,3]

Output

0

Test case 06:

Input

prices =[2,0,4,1,7]

Output

7

Test case 07:

Input

prices =[3,9]
Output
6

Test case 08:

Input
prices =[5]
Output
0

Test case 09:

Input
prices =[1000,5000,8000,9000,7000]
Output
8000

Test case 10:

Input
prices =[0,100,200,300,400]
Output
400

Test case 11:

Input
prices =[2,1,4,3,6,5,8,7]
Output
7

04.Problem

Instagram Account Spam Detection

Nisha categorizes an instagram account as spam, if the following count of the account is more than 10 times the count of followers.
Given the following and follower count of an account as X and Y respectively, find whether it is a spam account.

Input Format
The first line of input will contain a single integer T, denoting the number of test cases.
Each test case consists of two space-separated integers X and Y — the following and follower count of an account, respectively.

Output Format
For each test case, output on a new line, YES, if the account is spam and NO otherwise.

Test case 01:
Input:
3
20 2
15 1
30 3

Output:
YES
YES
YES

Test case 02:
Input:

2
10 5
8 2

Output:
NO
NO

Test case 03:
Input:
1
1 1

Output:
NO

Test case 05:
Input:
1
1000000000 100000000
Output:
YES

Test case 06:
Input:
4
25 2
12 4
18 3
9 1
Output:
YES
YES
YES

NO

05.Problem

Anagram Checker

Alice and Bob are avid Scrabble players who love challenging each other with word games. One day, Alice comes up with a new game where she checks if two words are anagrams of each other.

Given two strings s and t, return true if t is an anagram of s, and false otherwise.

Test case 01:

Input: s = "anagram", t = "nagaram"

Output: true

Test case 02:

Input: s = "rat", t = "car"

Output: false

Test case 03:

Input: s = "aab",t = "bba"

Output: false

Test case 04:

Input: s = "dormitory",t = "dirty room"

Output: true

Test case 05:

Input: s = "hello",t = "world"

Output: false

Test case 06:

Input: s = "listen",t = "silent"

Output: false

06.Problem

Longest Common Prefix Finder

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Test case 01:

Input: ["flower","flow","flight"]

Output: "fl"

Test case 02:

Input: ["dog","racecar","car"]

Output: ""

Test case 03:

Input: ["apple","appetizer","apricot"]

Output: "ap"

Test case 04:

Input: ["abc123","abc456","abc789"]

Output: "abc"

Test case 05:

Input: ["","world","water"]

Output: ""

Test case 06:

Input: ["hello"]

Output: "hello"

07.Problem

Last Word Length Finder

Sarah is a student who loves playing word games. Today, she was given a challenge by her friend Alex. Alex gave her a string containing words and spaces and asked her to find the length of the last word in the string.

Sarah, being a programming enthusiast, decided to write a function to solve this challenge. The function takes a string '**s**' as input and returns the length of the last word in the string.

Test case 01:
Input: s = "Hello World"
Output: 5

Test case 02:
Input: s = " fly me   to   the moon"
Output: 4

Test case 03:
Input: s = "luffy is still joyboy"
Output: 6

Test case 04:
Input: s = "luf"
Output: 3

Test case 05:
Input: s = "hi sarah"
Output: 5

Test case 06:
Input: s = "my computer"
Output: 8

08.Problem
Binary Tree Equality Checker

Emma and Liam, computer science enthusiasts, were exploring binary trees during their weekend coding session. Emma came up with an interesting challenge for Liam: checking if two binary trees(p&q) are the same or not.
if it's same → output: true
if it's not same → output: false

Test case 01:
Input: p = [1,2,3], q = [1,2,3]
Output: true

Test case 02:
Input: p = [1,2], q = [1,null,2]
Output: false

Testcase 03:
Input: p = [1,2,1], q = [1,1,2]
Output: false

Testcase 04:
Input: p = [1, 2, None, 3, None] ,q = [1, 2, None, 3, None]
Output:true

Test case 05:
Input:p = [1, 2, None, 3] ,q = [1, 2, 3, None]
output: false

Test case 06:
Input: p = [1, 2, 3, 4, 5],q = [1, 2, 3, 4, 5]
Output: true

09.Problem

Binary Tree Inorder Traversal

Olivia, an avid explorer of binary trees, loves navigating through the nodes in different orders. Today, she's curious about the inorder traversal of a binary tree and decides to return inorder traversal of its nodes values

Test case 01:
Input: [1,null,2,3]
Output: [1,3,2]

Test case 02
Input: [ ]
Output: [ ]

Test case 03
Input: [1]
Output: [1]

Test case 04
Input: [1, [2, [4, null, null], [5, null, null]], [3, null,null]]
Output: [4, 2, 5, 1, 3]

Test Case 05:
Input: [10, [5, [2, null, null], [8, [7, null, null], [9, null,null]]],[15,[12, null, null], [20, null, null]]]
Output: [2, 5, 7, 8, 9, 10, 12, 15, 20]

Test Case 06:
Input: [5, [3, [1, null, null], [4, null, null]], [7, [6, null, null], [8, null, null]]]
Output: [1, 3, 4, 5, 6, 7, 8]

10.Problem

Binary String Addition

Sophia and Ethan, both curious about binary numbers, decided to have some fun adding binary strings. Sophia provided Ethan with two binary strings, and together, they aimed to understand the result of adding them.

Constraints

- a and b consist only of '0' or '1' characters.
- Each string does not contain leading zeros except for the zero itself.

Test case 01:
Input: a = "11", b = "1"
Output: "100"

Test case 02:
Input: a = "1010", b = "1011"
Output: "10101"

Test case 03:
Input: a = "1010", b = "1011"
Output: "10101"

Test case 04:
Input: a = "1101", b = "1011"
Output: "11000"

Test case 05:
Input: a = "101", b = "111"
Output: "1100"

Test case 06:
Input: a = "00101", b = "111"
Output: "01000"

Test case 07:
Input: a = "0", b = "0"
Output: "0"

11.Problem

Biryani Cooking Class Cost Calculator

According to a recent survey, Biryani is the most ordered food. Vibi wants to learn how to make world-class Biryani from Jenny's lecture classes. Vibi will be required to attend Jenny's lecture classes for X weeks, and the cost of classes per week is Y coins. What is the total amount of money that Vibi will have to pay?

Input Format

The first line of input will contain an integer T — the number of test cases. The description of T test cases follows.The first and only line of each test case contains two space-separated integers X and Y, as described in the problem statement.

Output Format

For each test case, output on a new line the total amount of money that Vibi will have to pay.

Test case 01:
Input:

4
1 10
1 15
2 10
2 15

Output:
10
15
20
30

Test case 02
Input:
3
3 5
4 8
2 12
Output:
15
32
24

Test case 03
Input:
3
5 6
2 3
4 11
Output:
30
8
44

Test case 04
Input:
4
5 8
2 6
6 12
7 4
Output:
40
12
72
28

Test case 05
Input:
2
8 5
4 3

Output:
40
12

Test case 06
Input:
3
3 7
7 8
2 7
Output:
21
56

12.Problem

Ludo Token Entry Checker

Surya is playing Ludo. According to the rules of Ludo, a player can enter a new token into the play only when he rolls a 6 on the die.

In the current turn, Surya rolled the number X on the die. Determine if Surya can enter a new token into the play in the current turn or not.

Input Format

The first line contains a single integer T — the number of test cases. Then the test cases follow.

The first and only line of each test case contains one integer X — the number rolled by the Surya on the die.

Output Format

For each test case, output YES if the Surya can enter a new token in the game. Otherwise, output NO.

Test case 01:

Input

3

1

6

3

Output

NO

YES

NO

Test case 02:

Input:

5

1
3
5
2
4
Output:
NO
NO
NO
NO
NO

Test case 03:
Input
3
4
6
1
6
Output
NO
YES
NO
YES

Test case 04:
Input
2
5
2

Output
NO

NO

Test case 05:
Input
3
4
6
5
2
Output
NO
YES
NO
NO

Test case 06:
Input
3
6
2
1
4
Output
YES
NO
NO
NO

13.Problem

Textbook Reading Plan

Priyan has started studying for the upcoming test. The textbook has N pages in total. Priyan wants to read at most X pages a day for Y days.Find out whether it is possible for Priyan to complete the whole book.

Input Format

The first line of input will contain a single integer T, denoting the number of test cases.

The first and only line of each test case contains three space-separated integers N,X, and Y — the number of pages, the number of pages Priyan can read in a day, and the number of days.

Output Format

For each test case, output on a new line, YES, if Priyan can complete the whole book in given time, and NO otherwise.

Test case 01

Input

4

5 2 3

10 3 3

7 7 1

3 2 1

Output

YES

NO

YES

NO

Test case 02
Input:
2
120 15 10
200 8 30

Output
NO
YES

Test case 03
Input
3
80 20 4
60 10 7
150 5 35
Output
YES
YES
NO

Test case 04
Input
2
15 4 4
20 2 10
Output
YES
YES

Test case 05

Input
3
8 2 3
12 3 6
6 1 5
Output
NO
YES
NO

Test case 06
Input
1
25 5 5
Output
YES

14.Problem
Gas Station Circuit Journey
There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].

You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from the ith station to its next (i + 1)th station.

You begin the journey with an empty tank at one of the gas stations.

Given two integer arrays **gas** and **cost**, return the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return -1. If there exists a solution, it is guaranteed to be unique

Test case 01
Input: gas = [1,2,3,4,5], cost = [3,4,5,1,2]
Output: 3

Test case 02
Input: gas = [2,3,4], cost = [3,4,3]
Output: -1

Test case 03
Input: gas = [1, 1, 3, 1],cost = [2, 2, 1, 1]
Output: 2

Test case 04
Input:gas = [4, 5, 2, 6, 5, 3],cost = [3, 2, 7, 3, 2, 9]
Output: 3

Test case 05
Input:gas = [5, 8, 2, 8],cost = [6, 5, 6, 6]
Output:3

Test case 06
Input:gas =[3, 1, 2] , cost = [1,2,3]
Output:0

15.Problem

Binary Transformation

Given an integer n, you must transform it into 0 using the following operations any number of times:

- Change the rightmost (0^{th}) bit in the binary representation of n.
- Change the ith bit in the binary representation of n if the $(i-1)^{th}$ bit is set to 1 and the $(i-2)^{th}$ through 0^{th} bits are set to 0.

Return the minimum number of operations to transform n into 0.

Test case 01
Input: n = 3
Output: 2

Test case 02
Input: n = 6
Output: 4

Test case 03
Input: n = 1001
Output: 689

Test case 04
Input: n = 10000
Output: 14879

Test case 05
Input: n = 30
Output: 60

Test case 06
Input: n = 16
Output: 31

16.Problem

Remove Duplicates and Count Unique Elements

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums.

Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:

Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of nums.

Return k.(IF INPUT ALSO IN CHAR TYPE)

Test case 01
Input:[0,0,1,1,1,2,2,3,3,4]
Output:[0,1,2,3,4]

Test case 02
Input :[1,1,2]
Output:[1,2]

Test case 03
Input :[A,C,C,G,H,E,E,E,E,T,Y,U]
Output:[A,C,G,H,E,T,Y,U]

Test case 04
Input :[0000000000000,123,12]
Output:[0,123,12]

Test case 05
Input: [1, 1, 2, 2, 2, 3, 4, 4, 4, 5, 5, 6, 6, 7, 8, 9, 9, 10, 11, 11, 12, 12, 12, 13]
Output:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

Test case 06
Input: nums = [1, 1, 1, 1, 1]
Output:[1]

17 . PROBLEM

Pretty Numbers Counter

Vishnu likes the number 239 . Therefore, he considers a number pretty if its last digit is 2 ,3 or 9.
Vishnu wants to watch the numbers between L and R (both inclusive), so he asks you to determine how many pretty numbers are in this range. Can you help him?

Input

- The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows.
- The first and only line of each test case contains two space-separated integers L and R.

Output

For each test case, print a single line containing one integer — the number of pretty numbers between L and R.

Test case 01

Input
2
12345 67890
11 33

Output
16663
8

Test case 02
Input:
2
10 20
25 35
Output:
2
3

Test case 03
Input:
3
500 600
238 242
1 1
Output:
30
2
0

Test case 04
Input:
1

500 600

Output:
30

Test case 05
Input:
1
238 242
Output:
2

Test case 06
Input:
1
888 900
Output:
4

18 .Problem
Lucky Letter Finder
Rethanya considers the number 7 lucky. As a result, she believes that the 7-th letter she sees on a day is his lucky letter of the day.You are given a string S of length 10, denoting the first 10 letters Rethanya saw today.(Both integer and char type input)
What is Rethanya 's lucky letter?
Input Format
The only line of input contains a string S, of length 10.

Output Format

Print a single character: Rethanya's lucky letter .

Test case 01
Input: proceeding
Output:d

Test case 02
Input: limiting
Output: n

Test case 03
Input
ABCDEF777G
Output:
7

Test case04
Input
345678900
Output:
9

Test case 05
Input
XYZABC1234
Output
C

Test case 06
Input

XYBC1234
Output
3

19. Problem

Burger Making Challenge

Shri is fond of burgers and decided to make as many burgers as possible.

Shri has A patties and B buns. To make 1 burger, Shri needs 1 patty and 1 bun.

Find the maximum number of burgers that Shri can make.

Input Format

- The first line of input will contain an integer T — the number of test cases. The description of T test cases follows.
- The first and only line of each test case contains two space-separated integers A and B, the number of patties and buns respectively.

Output Format

For each test case, output the maximum number of burgers that Shri can make.

Test case 01

Input

4

2 2

2 3

3 2

23 17

Output
2
2
2
17

Test case 02
Input
3
5 3
2 8
10 10

Output
3
2
10

Test case 03
Input
3
7 4
9 5
1 16

Output
4
5
1

Test case 04
Input

1 1
Output
1

Test case 05
Input
1 0
Output
0

Test case 06
Input
0 0
Output
0

20. Problem
Kitchen Working Hours
The working hours of priya's kitchen are from X pm to Y pm ($1 \leq X < Y \leq 12$).
Find the number of hours priya works.
Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two space-separated integers X and Y — the starting and ending time of working hours respectively.

Output Format

For each test case, output on a new line, the number of hours priya works.

Test case 01

Input

4
1 2
3 7
9 11
2 10

Output

1
4
2
8

Test case 02

Input

3
5 10
8 2
12 3

Output

5
6
3

Test case 03

Input

3
6 9
8 10

12 4
Output
3
2
4

Test case 04
Input
3
7 9
9 10
12 2
Output
2
1
2

Test case 05
Input
2
5 9
8 11

Output
4
3

Test case 03
Input
3
5 10
6 8

11 4
Output
5
2
5

21. Problem

Binary's Hearing Range Checker

Siva's dog binary hears frequencies starting from 67 Hertz to 45000 Hertz (both inclusive).

If Siva's commands have a frequency of X Hertz, find whether binary can hear them or not.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of a single integer X - the frequency of Chef's commands in Hertz.

Output Format

For each test case, output on a new line YES, if binary can hear siva's commands. Otherwise, print NO.

Test case 01

Input

5
42
67
402
45000

45005

Output
NO
YES
YES
YES
NO

Test case 02
Input
3
100
5000
45001
Output
YES
YES
NO

Test case 03
Input
3
1
700
45050

Output
NO
YES
NO

Test case 04
Input

3
7000
50000
45050

Output
NO
NO
NO

Test case 05
Input
2
60
407

Output
NO
YES

Test case 06
Input
4
107
9001
45050
505

Output
YES
YES
NO
YES

24.Problem

Tom and Jerry Chase

In a classic chase, Tom is running after Jerry as Jerry has eaten Tom's favourite food.Jerry is running at a speed of X metres per second while Tom is chasing him at a speed of Y metres per second. Determine whether Tom will be able to catch Jerry.

Note that initially Jerry is not at the same position as Tom.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two space-separated integers X and Y — the speeds of Jerry and Tom respectively.

Output Format

For each test case, output on a new line, YES, if Tom will be able to catch Jerry. Otherwise, output NO.

Test case 01:

Input

4

2 3
4 1
1 1
3 5
Output
NO
YES
YES
NO

Test case 02:
Input
4
7 3
3 1
4 8
2 5
Output
YES
NO
NO
NO

Test case 03:
Input
3
5 3
4 5
10 8

Output
YES
NO

YES

Test case 04:
Input
3
7 3
9 5
9 8

Output
YES
YES
YES

Test case 05:
Input
4
6 3
4 9
12 8
6 2

Output
YES
NO
YES
YES

Test case 06:
Input
2
9 11
7 4

Output

NO
YES

25. Problem

Concert Attendance Decision

Four friends want to attend a concert. Each ticket costs X rupees.They have decided to go to the concert if and only if the total cost of the tickets does not exceed 1000 rupees. Determine whether they will be going to the concert or not.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of a single integer X, the cost of each ticket.

Output Format

For each test case, output YES if they will be going to the concert, NO otherwise.

Test case 01:
Input
4
100
500
250
1000

Output
YES
NO
YES
NO

Test case 02:
Input
3
200
250
300

Output
YES
YES
NO

Test case 03:
Input
3
2000
270
800

Output
NO
NO
NO

Test case 04:
Input
4
150

210
500
100

Output
YES
YES
NO
YES

Test case 05:
Input
3
290
250
300

Output
NO
YES
NO

Test case 06:
Input
3
120
230
390

Output
YES
YES
NO

26. Problem

Speeding Fine Calculator

Anu was driving on a highway at a speed of X km/hour.To avoid accidents, there are fine imposed on overspeeding as follows:

- No fine if the speed of the car ≤70 km/hour.
- Rs 500 fine if the speed of the car is strictly greater than 70 and ≤100.
- Rs 2000 fine if the speed of the car is strictly greater than 100.

Determine the fine Anu needs to pay.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of a single integer X denoting the speed of Anu's car.

Output Format

For each test case, output the fine paid by Anu.

Test case 01:

Input

7
40
110
70
100
69
101
85

Output
0
2000
0
500
0
2000
500

Test case 02:
Input
3
60
80
110

Output
0
500
2000

Test case 03:
Input
5
50
95
125
89
115

Output
0

500
500
2000
500
2000

Test case 04:
Input
4
60
80
110
75

Output
0
500
2000
500

Test case 05:
Input
2
70
108

Output
0
2000

Test case 06:
Input
3
65

89
120

Output
0
500
2000

26. Problem
You are given an integer n, the number of teams in a tournament that has strange rules:

- If the current number of teams is even, each team gets paired with another team. A total of n / 2 matches are played, and n / 2 teams advance to the next round.
- If the current number of teams is odd, one team randomly advances in the tournament, and the rest gets paired. A total of (n - 1) / 2 matches are played, and (n - 1) / 2 + 1 teams advance to the next round.
- Return the number of matches played in the tournament until a winner is decided.

Test case 01:
Input: n = 7
Output: 6

Test case 02:
Input: n =14
Output: 13

Test case 03:
Input: n =8

Output: 7

Test case 04:

Input: n =6

Output: 5

Test case 05:

Input: n = 4

Output: 3

Test case 06:

Input: n =181

Output: 180

1.Write a function to perform basic string compression using the counts of repeated characters. If the compressed string is not smaller than the original, return the original string.

Test Cases:

"aabccccccaaa" → "a2b1c5a3"

"abcdef" → "abcdef"

"aa" → "a2"

"aab" → "aab"

"abb" → "abb"

"aaaaabbbcc" → "a5b3c2"

"" → ""

"xyzzzz" → "x1y1z4"

"aabbaa" → "a2b2a2"

"ab" → "ab"

2.Write a function to check if a given number is an Armstrong number.

Test Cases:

153 → True

9474 → True

1634 → True

371 → True

8208 → True

163 → False

947 → False

407 → True

111 → False

1634 → True

3. Count Occurrences of a Character
Write a function to count the occurrences of a specific character in a given string.

Test Cases:

String: "programming", Character: 'g' → 2

String: "hello world", Character: 'o' → 2

String: "Python is fun", Character: 'i' → 2

String: "banana", Character: 'a' → 3

String: "mississippi", Character: 's' → 4

String: "abcdefg", Character: 'z' → 0

String: "testing", Character: 't' → 2

String: "racecar", Character: 'r' → 3

String: "abcdabcd", Character: 'd' → 2

String: "programming is interesting", Character: 'm' → 2

4.Remove Duplicates from a List
Write a function to remove duplicates from a list while maintaining the order.

Test Cases:

[1, 2, 2, 3, 4, 4, 5] → [1, 2, 3, 4, 5]

[10, 20, 10, 30, 40, 30] → [10, 20, 30, 40]

[5, 5, 5, 5, 5] → [5]

[1, 2, 3, 4, 5] → [1, 2, 3, 4, 5]

[10, 20, 30, 40, 50] → [10, 20, 30, 40, 50]

[1, 1, 1, 2, 2, 3, 3] → [1, 2, 3]

[5, 4, 3, 2, 1] → [5, 4, 3, 2, 1]

[7, 7, 7, 7] → [7]

[8, 8, 8, 8, 8, 8, 8] → [8]

[3, 1, 4, 1, 5, 9, 2, 6] → [3, 1, 4, 5, 9, 2, 6]

5.Count Denominations
Ganesh wants to find the number of denominations for a given amount. The possible denominations are 1000, 500, 100, 50, 10, 1. Help him to find the solution.

Test Cases:

Input: 2350 → Output: 4

Input: 750 → Output: 2

Input: 12345 → Output: 8

Input: 5000 → Output: 5

Input: 9876 → Output: 7

Input: 100 → Output: 1

Input: 999 → Output: 7

Input: 1 → Output: 1

Input: 2468 → Output: 6

Input: 888 → Output: 5

6.Find the product of the digits in a given number

Input format :
Integer

Output format :
Integer

Testcase:
Input: 12345 → Output: 120 (1 * 2 * 3 * 4 * 5)

Input: 9876 → Output: 3024 (9 * 8 * 7 * 6)

Input: 54321 → Output: 120 (5 * 4 * 3 * 2 * 1)

Input: 111 → Output: 1 (1 * 1 * 1)

Input: 876543 → Output: 6048 (8 * 7 * 6 * 5 * 4 * 3)

Input: 999 → Output: 729 (9 * 9 * 9)

Input: 456 → Output: 120 (4 * 5 * 6)

Input: 789 → Output: 504 (7 * 8 * 9)

Input: 100 → Output: 0 (1 * 0 * 0)

Input: 111111 → Output: 1 (1 * 1 * 1 * 1 * 1 * 1)

7. Least Frequent Digit

Rithish wants to find which digit occurs the least number of times among the given set of values. Help him to solve the problem.

Input format:

First line contains the size of the array.

Second line contains the array elements.

Output format:

Print the digit that occurs the least number of times.

Code constraints:
The array elements are integers.

Test Cases:
5
12345

8
11223344

6
9876543

7
9999888

4
7777

9
555555555

10
1234567890

5

12121

7
9876543

6
999999

8.Given an array arr[] and a number K. The task is to find the count of the element having odd and even number of set-bits after taking XOR of K with every element of the given arr[].
Input format:
First line contains the array size.
Second line contains the array elements.
Third line contains the K value. Get me 10 testcase.

Test Case 1:
Input:
5
10 20 15 2 5
8

Test Case 2:
Input:
3
1 2 3
2

Test Case 3:
Input:
4
1 5 7 3
6

Test Case 4:
Input:
5
10 20 30 40 50
6

Test Case 5:
Input:
4
1 7 3 5
2

Test Case 6:
Input:
5
10 25 15 5 25
8

Test Case 7:
Input:

3
2 4 6
3

Test Case 8:

Input:

4
5 10 15 20
7

Test Case 9:
Input:
5
1 5 9 13 17
10

Test Case 10:
Input:
4
10 20 30 40
15

9.Given a string s, find the length of the longest substring

without repeating characters.

TestCase:

Input	Output
"abcabcbb"	4
"bbbbb"	2
"pwwkew"	3
"aab"	3
"dvdf"	4
"tmmzuxt"	5
"ohvhjdml"	6
"Nndfnasd"	5
"aaaaaaaaaa"	2
"abcdefghijklmnopqrstuvwxyz"	26

10.You are given two integer arrays nums1 and nums2 of lengths m and n respectively. nums1 and nums2 represent the digits of two numbers. You are also given an integer k. Create the maximum number of length k <= m + n from digits of the two numbers. The relative order of the digits from the same array must be preserved.
Return an array of the k digits representing the answer.

Testcase:

nums1	nums2	k	Output
[1,3,4,5]	[2,6,7]	3	[5,7,6]
[9,8,7,6,5]	[2,1]	5	[9,8,7,6,2]
[1,2,3,4]	[0,5,6]	5	[5,4,3,2,1]
[6,7]	[6,7,8,9]	3	[9,7,6]
[2,3,4]	[9,8,5,7]	4	[9,8,7,4]
[5,9,8,6]	[2,3,1]	4	[9,8,6,5]
[8,9,7]	[4,5,6,2]	4	[9,8,7,6]
[2,3,9,8]	[1,4,5,6,7]	9	[9,8,7,6,5,4,3,2]
[7,5,3,1]	[8,9,2,4,6]	5	[9,8,7,5,3]
[6,7,8,9]	[1,2,3,4]	6	[9,8,7,6,5,4]

11.A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. A divisor of an integer x is an integer that can divide x evenly. Given an integer n, return true if n is a perfect number, otherwise return false.

TESTCASE:

Input	Output
6	True
28	True
496	True
8128	True
33550336	True
5	False
8	False
9	False

24 False

100 False

12.
Sure, here is the problem statement for "Minimum Number of Groups to Create a Valid Assignment" along with 10 test cases:

Problem Statement:

You are given a zero-indexed array nums of length n. You need to assign n students to groups such that the following conditions are met:

All students must be assigned to a group.
Each group must have at least two students.
The difference in the number of students between any two groups should not be more than one.
Return the minimum number of groups required to satisfy all the given conditions.

Test Cases:

Input	Output
[3, 2, 1, 1, 1, 2, 2, 1]	4
[4, 1, 2, 4, 1, 3]	3
[1, 1, 1, 1, 1]	1
[1, 2, 3, 4, 5]	5
[2, 1, 2, 1, 2, 1]	3
[1, 1, 2, 2, 2, 2]	3
[1, 2, 1, 2, 1, 2]	2
[2, 2, 1, 1, 1, 2]	3
[1, 2, 3]	2
[2, 1, 1, 1, 2, 2]	3

13.You are given a 0-indexed integer array nums. You have to find the maximum sum of a pair of numbers from nums such that the maximum digit in both numbers are equal.

testcase:

Input	Output
[1, 4, 2, 3]	0
[5, 4, 2, 1]	0
[3, 5, 4, 2]	7
[2, 5, 4, 1]	6
[1, 5, 4, 2]	6
[2, 4, 5, 1]	7
[5, 1, 4, 2]	0

[4, 5, 1, 2] 9

[2, 1, 5, 4] 7

[1, 2, 5, 4] 6

14.
You are given a 0-indexed 2D integer matrix grid of size 3 * 3, representing the number of stones in each cell. The grid contains exactly 9 stones, and there can be multiple stones in a single cell.
In one move, you can move a single stone from its current cell to any other cell if the two cells share a side.

Return the minimum number of moves required to place one stone in each cell.

testcase:

Input (grid)	Output
[[1, 1, 0], [1, 1, 1], [1, 2, 1]]	3
[[0, 2, 2], [1, 1, 1], [2, 1, 0]]	4
[[1, 0, 2], [0, 2, 1], [2, 1, 0]]	4
[[2, 0, 1], [1, 2, 1], [0, 1, 2]]	3
[[0, 1, 2], [2, 1, 0], [1, 0, 2]]	4
[[2, 1, 0], [0, 2, 1], [1, 0, 2]]	3
[[1, 2, 0], [0, 1, 2], [2, 0, 1]]	3
[[0, 2, 1], [1, 0, 2], [2, 1, 0]]	3
[[1, 0, 2], [2, 1, 0], [0, 2, 1]]	3
[[2, 0, 1], [0, 2, 1], [1, 0, 2]]	3

15.Given a 0-indexed string s, permute s to get a new string t such that:

All consonants remain in their original places. More formally, if there is an index i with 0 <= i < s.length such that s[i] is a consonant, then t[i] = s[i].
The vowels must be sorted in the nondecreasing order of their ASCII values. More formally, for pairs of indices i, j with 0 <= i < j < s.length such that s[i] and s[j] are vowels, then t[i] must not have a higher ASCII value than t[j].

Testcase:

Input (s)	Output (t)
"abcabcbb"	"abcabcbb"
"leetcode"	"leeaetcod"
"aiieou"	"aeiiou"
"hello"	"hello"
" leetcode"	" leeaetcod"
"aiieou"	"aeiiou"
" hello"	" hello"
" leetcode"	" leeaetcod"
"aiieou"	"aeiiou"
" hello"	" hello"

16.
You are given a 0-indexed integer array arr, and an m x n integer matrix mat. arr and mat both contain all the integers in the range [1, m * n].
Go through each index i in arr starting from index 0 and paint the cell in mat containing the integer arr[i].

testcase:

Input (arr, mat)	Output
([1, 3, 4, 2], [[1, 4], [2, 3]])	2
([2, 8, 7, 4, 1, 3, 5, 6, 9], [[3, 2, 5], [1, 4, 6], [8, 7, 9]])	3
([5, 4, 2, 1], [[1, 4], [2, 3]])	-1
([3, 2, 1, 1, 1, 2, 2, 1], [[3, 2, 5], [1, 4, 6], [8, 7, 9]])	-1
([1, 4, 2], [[4, 1], [2, 3]])	2
([2, 8, 7, 4, 1, 3, 5, 6, 9], [[1, 2, 3], [4, 5, 6], [7, 8, 9]])	3
([1, 5, 4, 2], [[4, 1], [2, 3]])	2
([3, 5, 4, 2], [[4, 1], [2, 3]])	2
([2, 1, 5, 4], [[4, 1], [2, 3]])	-1
([1, 2, 5, 4], [[4, 1], [2, 3]])	-1

17.You are given a 0-indexed m x n matrix grid consisting of positive integers.

You can start at any cell in the first column of the matrix, and traverse the grid in the following way:
From a cell (row, col), you can move to any of the cells: (row - 1, col + 1), (row, col + 1) and (row + 1, col + 1) such that the value of the cell you move to, should be strictly bigger than the value of the current cell.

testcase:

Input (grid)	Output
[[4, 3, 2, 1], [7, 6, 5, 4], [8, 2, 2, 3]]	4
[[5, 4, 3, 2], [6, 5, 4, 1], [7, 6, 5, 4]]	5
[[3, 2, 1, 1, 1, 2, 2, 1], [4, 1, 2, 4, 1, 3], [5, 3, 2, 5, 4, 4]]	3
[[2, 1, 2, 1, 2, 1], [3, 2, 2, 3, 2, 2], [4, 3, 3, 4, 3, 3]]	4
[[1, 1, 2, 1, 2, 1], [2, 2, 3, 2, 3, 2], [3, 3, 3, 3, 3, 3]]	3
[[1, 2, 3, 4, 5], [2, 3, 4, 5, 6], [3, 4, 5, 6, 7], [4, 5, 6, 7, 8]]	7
[[5, 4, 3, 2], [6, 5, 4, 1], [7, 6, 5, 4]]	5
[[3, 2, 1, 1, 1, 2, 2, 1], [4, 1, 2, 4, 1, 3], [5, 3, 2, 5, 4, 4]]	3
[[2, 1, 2, 1, 2, 1], [3, 2, 2, 3, 2, 2], [4, 3, 3, 4, 3, 3]]	4
[[1, 1, 2, 1, 2, 1], [2, 2, 3, 2, 3, 2], [3, 3, 3, 3, 3, 3]]	3

18.You are given an integer array banned and two integers n and maxSum. You are choosing some number of integers following the below rules:
The chosen integers have to be in the range [1, n].
Each integer can be chosen at most once.
The chosen integers should not be in the array banned.

The sum of the chosen integers should not exceed maxSum.

Return the maximum number of integers you can choose following the mentioned rules.

Testcase:

Input	Output
banned = [1, 6, 5], n = 5, maxSum = 6	2
banned = [1, 2, 3, 4, 5, 6, 7], n = 8, maxSum = 1	0

banned = [11], n = 7, maxSum = 50 7

banned = [7, 3, 4, 8], n = 10, maxSum = 16 3

banned = [9, 8, 7, 3, 4, 5, 10, 6], n = 10, maxSum = 20 5

banned = [4, 2, 1], n = 10, maxSum = 6 3

banned = [3, 4, 1, 5, 9, 2, 6, 8, 7], n = 10, maxSum = 20 6

banned = [5, 8, 4, 6, 7, 3, 1, 2, 9], n = 10, maxSum = 20 7

banned = [2, 7, 4, 9, 5, 8, 1, 6, 3], n = 10, maxSum = 20 8

banned = [6, 2, 3, 8, 7, 9, 4, 5, 1], n = 10, maxSum = 20 8

19.Given a positive integer num, split it into two non-negative integers num1 and num2 such that:

The concatenation of num1 and num2 is a permutation of num.
In other words, the sum of the number of occurrences of each digit in num1 and num2 is equal to the number of occurrences of that digit in num.

num1 and num2 can contain leading zeros,
Return the minimum possible sum of num1 and num2.

testcases:

Input	Output
num = 1234	1354
num = 11223	11321
num = 121234	112342
num = 112334	113421
num = 112345	112435
num = 123456	123546
num = 132456	124356
num = 142356	123546
num = 152346	124356
num = 162345	123546

20.
You are given a positive integer n. Each digit of n has a sign according to the following rules:

The most significant digit is assigned a positive sign.Each other digit has an opposite sign to its adjacent digits.
Return the sum of all digits with their corresponding sign.

Input (n)	Output
12345	4
54321	-4
1001	1
1010	2
1011	-1
1020	2
1021	0
1022	-2
1023	-1
1030	3

21.
You are given a 0-indexed array of string words and two integers left and right.
A string is called a vowel string if it starts with a vowel character and ends with a vowel character where vowel characters are 'a', 'e', 'i', 'o', and 'u'.

Return the number of vowel strings words[i] where i belongs to the inclusive range [left, right].

Input	Output
words = ["a", "e", "i"], left = 0, right = 2	3
words = ["aba", "bcb", "ece", "aa", "e"], left = 0, right = 2	3
words = ["aba", "bcb", "ece", "aa", "e"], left = 1, right = 4	3
words = ["aba", "bcb", "ece", "aa", "e"], left = 1, right = 1	0
words = ["aba", "ece"], left = 0, right = 1	2
words = ["aba", "ece"], left = 0, right = 0	1
words = ["aba", "ece", "bcb"], left = 1, right = 2	1
words = ["aba", "ece", "bcb"], left = 2, right = 2	1

words = ["aba", "ece", "bcb"], left = 2, right = 0	0
words = ["aba", "ece", "bcb"], left = 0, right = 2	3

22.You are given two 0-indexed integer arrays nums1 and nums2 of equal length n and a positive integer k. You must choose a subsequence of indices from nums1 of length k.
For chosen indices i0, i1, ..., ik - 1, your score is defined as:

The sum of the selected elements from nums1 multiplied with the minimum of the selected elements from nums2.

It can defined simply as: (nums1[i0] + nums1[i1] +...+ nums1[ik - 1]) * min(nums2[i0] , nums2[i1], ... ,nums2[ik - 1]).

TESTCASE:

Input (nums1, nums2, k)	Output
nums1 = [1], nums2 = [1], k = 1	1
nums1 = [2, 3, 4], nums2 = [1, 2, 3], k = 2	4
nums1 = [5, 1, 2, 3, 4], nums2 = [3, 2, 1, 4, 5], k = 3	15
nums1 = [4, 2, 3, 1, 5], nums2 = [5, 3, 1, 4, 2], k = 4	10
nums1 = [5, 3, 1, 4, 2], nums2 = [2, 1, 4, 5, 3], k = 3	9
nums1 = [2, 3, 4, 1, 5], nums2 = [5, 4, 3, 2, 1], k = 4	8
nums1 = [1, 2, 3, 4, 5], nums2 = [2, 3, 4, 1, 5], k = 3	12
nums1 = [1, 5, 2, 3, 4], nums2 = [4, 3, 2, 1, 5], k = 3	15
nums1 = [3, 2, 1, 4, 5], nums2 = [2, 1, 4, 5, 3], k = 4	10
nums1 = [4, 2, 1, 3, 5], nums2 = [4, 3, 1, 2, 5], k = 3	6

23.You are given an m x n integer matrix grid.We define an hourglass as a part of the matrix with the following form:Return the maximum sum of the elements of an hourglass.
Note that an hourglass cannot be rotated and must be entirely contained within the matrix.

Input (grid)	Output
[[6, 2, 1, 3], [4, 2, 1, 5], [9, 2, 8, 7], [4, 1, 2, 9]]	30
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	35
[[3, 2, 1, 1, 1, 2, 2, 1], [4, 1, 2, 4, 1, 3], [5, 3, 2, 5, 4, 4]]	33

[[2, 1, 2, 1, 2, 1], [3, 2, 2, 3, 2, 2], [4, 3, 3, 4, 3, 3]]	45
[[1, 1, 2, 1, 2, 1], [2, 2, 3, 2, 3, 2], [3, 3, 3, 3, 3, 3]]	48
[[7, 7, 5, 7], [1, 1, 4, 1], [1, 1, 5, 1], [7, 7, 4, 7]]	44
[[6, 2, 1, 3], [4, 2, 1, 5], [9, 2, 8, 7], [4, 1, 2, 9]]	30
[[3, 2, 1, 1, 1, 2, 2, 1], [4, 1, 2, 4, 1, 3], [5, 3, 2, 5, 4, 4]]	33
[[2, 1, 2, 1, 2, 1], [3, 2, 2, 3, 2, 2], [4, 3, 3, 4, 3, 3]]	45
[[1, 1, 2, 1, 2, 1], [2, 2, 3, 2, 3, 2], [3, 3, 3, 3, 3, 3]]	48

24.You are given an array nums consisting of positive integers.You have to take each integer in the array, reverse its digits, and add it to the end of the array. You should apply this operation to the original integers in nums.Return the number of distinct integers in the final array.

testcase:

Input (nums)	Output
nums = [123, 321, 456]	3
nums = [234, 123, 456, 789]	4
nums = [543, 234, 789, 123]	4
nums = [456, 789, 123, 234]	4
nums = [789, 456, 234, 123]	4
nums = [12345, 54321, 98765]	3
nums = [54321, 12345, 98765]	3
nums = [98765, 54321, 12345]	3
nums = [12345, 98765, 54321]	3
nums = [54321, 98765, 12345]	3

25.You are given an integer array cookie, where cookies[i] denotes the number of cookies in the ith bag. You are also given an integer k that denotes the number of children to distribute all the bags of cookies to. All the cookies in the same bag must go to the same child and cannot be split up.The unfairness of a distribution is defined as the maximum total cookies obtained by a single child in the distribution.Return the minimum unfairness of all distributions.

TESTCASE:

Input (cookies, k)	Output
cookies = [7, 3, 2, 5, 4], k = 3	2
cookies = [10, 7, 5, 3, 2], k = 3	5
cookies = [12, 4, 5, 3, 8], k = 3	8
cookies = [15, 7, 10, 3, 2], k = 3	7
cookies = [18, 9, 12, 5, 2], k = 3	10
cookies = [8, 4, 5, 3, 1], k = 2	5
cookies = [11, 6, 7, 3, 2], k = 2	7
cookies = [14, 8, 9, 4, 3], k = 2	8
cookies = [17, 10, 11, 5, 4], k = 2	10
cookies = [20, 12, 13, 6, 5], k = 2	11