Chapter 0

Coding/Programming Problems for Campus Placements

The book in your hand is the only textbook that gives you exposure regarding the kind of problem asked and the format in which they are asked during the placement process by the majority of the IT companies. You are required to strictly adhere to the instructions otherwise your submission of the solution can be rejected.

The problems listed here, are just indicative, and will be expanded further in future editions. The problems listed and similar had been asked in the previous placements in the coding/programming section of the first phase (screening) by the majority of the IT companies including TCS, Wipro, IBM, Microsoft, Infosys, Cognizant, Accenture, Amazon, Facebook, etc. While going through the content of the book, try to solve these problems, and this practice will help you to harness your problem-solving and coding skills to crack the placement with flying colors.

As a special gesture, the readers of this book will be provided with placement assistance free of cost, for which you may be required to register online.

Best wishes for your great future.

1. Area of Circle

Write a program to find area of circle.

Input Format:
Input consists of float value which corresponds to radius of circle.

Output Format:
Output consist of area of circle in mtsq correct to 2 decimal places.

Sample Input:

5.5
Sample Output:
94.99

2. Leap Year

A year will be a leap year if it is divisible by 4 but not by 100. If a year is divisible by 4 and by 100, it is not a leap year unless it is also divisible by 400.

Write a program to check whether a given year is a leap year or not.

Input Format:
Input consists of a single integer.

Output Format:
Output consists of a single line. Refer sample output for details.

Sample Input 1:
2020
Sample Output 1:
2020 is a leap year

Sample Input 2:
2019
Sample Output 2:
2019 is not a leap year

3. Validity of Date

Write a program to check whether the given date in format *dd/mm/yyyy* is valid or not.

Input Format:
Input consists of date in given format.

Output Format:
Output consists of a single line. Refer sample output for details.

Sample Input 1:
Enter date in format dd/mm/yyyy:
19/11/2020
Sample Output 1:
19/11/2020 is valid date.

Sample Input 2:
Enter date in format dd/mm/yyyy:
29/02/2019
Sample Output 2:
29/02/2019 invalid date.

4. Find Your Day of Birth

Write a program to find the day on which you were born. Your date-of-birth is given in the format *dd/mm/yyyy*.

Input Format:
Input consists of date in given format.

Output Format:
Output consists of a single line. Refer sample output for details.

Sample Input 1:
Enter your date of birth dd/mm/yyyy:
28/10/1955

Sample Output 1:
You was born on Friday.

**Bill Gate, Founder of Microsoft, was born on this day.*

5. Armstrong Number

An Armstrong number is a 3-digit integer such that the sum of the cubes of its digits is equal to the number itself.

For example, 371 is an Armstrong number, since

3^3 + 7^3 + 1^3 = 371

Write a program to find whether a given 3-digit number is an Armstrong number or not. Restrict your input to 3-didit number otherwise display message "Wrong input".

Input Format:
Input consists of a single integer.

Output Format:
Refer sample output for details.

Sample Input 1:
153
Sample Output 1:
Armstrong Number

Sample Input 2:
135
Sample Output 2:
Not an Armstrong Number

6. Prime Number

A prime number is a natural number that is divisible by only 1 and itself only.

Write a program to find whether the given natural number 'n' is a prime number or not.

Input Format:
Input consists of a single integer.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter the given natural number: 17
15 is a prime number

Sample Input and Output 2:
Enter the given natural number: 145
145 is not a prime number

7. Sum of digits

Write a program to find the sum of digits of a given natural number 'n'.

Input format:
Input consists of an integer value.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter given natural number: **1345**
Sum of digits in 1345 is 13.

Sample Input and Output 2:
Enter given natural number: **1111111111**
Sum of digits in **1111111111** is 10.

8. Factorial

Write a program to find the factorial of a given natural number 'n'.

Input format:
Input consists of an integer value.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter given natural number: 4
4! = 24

Sample Input and Output 2:
Enter given natural number: 6
6! = 720

9. Palindrome Number

A Palindrome number is one which reads same from both the ends. For example, 1881, 232454.

Write a program to find whether the given number 'n' is a palindrome or not.

Input Format:
Input consists of a single integer.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter the given natural number: 1001
1001 is a palindrome.

Sample Input and Output 2:
Enter the given natural number: 1100
1100 is not a palindrome.

10. Fibonacci Sequence

Write a program to print first 'n' terns of the Fibonacci sequence.

For example, first 7 terms of Fibonacci sequence

0 1 1 2 3 5 8

Input format:
Input consists of an integer value.

Output Format:
Refer sample output for details.

Sample Input and Output:
Enter value for n: **10**
First 10 terms of Fibonacci sequence are
0 1 1 2 3 5 8 13 21 34

11. Perfect Number

A perfect number is a number in which the sum of its proper divisors is equal to the number itself. Proper divisors of a number are all divisors of a number excluding itself.

Write a program to find whether a given number 'n' is a perfect number or not.

Input format:
Input consists of an integer value.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter given natural number: 6
6 is a perfect number.
Sample Input and Output 2:
Enter given natural number: 12
12 is not a perfect number.

12. Strong Number

A strong number is a number whose sum of the factorial of its each digit equals to the number itself.

For example, 145 is a strong number since

1! + 4! + 5! = 1 + 24 + 120 = 145

Write a program to find whether a given number 'n' is a strong number or not.

Input Format:
Input consists of a single integer.

Output Format:
Refer sample output for details.

Sample Input and Output 1:
Enter the given number: 145
145 is a strong number.

Sample Input and Output 2:
Enter the given number: 110
110 is not a strong number.

13. Special Number

A 2-digit number is said to be a special number if the sum of the sum of its digits and the products of its digits is equal to the number itself.

For example, 19 is a special number. The digits in 19 are 1 and 9. The sum of the digits is 10 and the product of the digits is 9, sum of sum of digits and product of digits is 19.

Write a program to find all special numbers.

Input Format:
Nil

Output Format:
List of all special numbers separated by one or more spaces. Refer to sample output for more details.
Output:
Here is the list of special numbers.
19 29 39 49 59 69 79 89 99

14. GCD of numbers

Write a program to find the greater common divisor (GCD)/highest common factor (HCF) of two natural numbers 'm' and 'n'.

Input Format:
Input consists of 2 natural numbers.

Output Format:
Refer sample input and output for formatting specifications.

Sample Input and Output:
Enter first natural number 'm' : 36
Enter second natural number 'n' : 27

GCD(36,27) = 9

15. LCM of numbers

Write a program to find the least common multiplier (LCM) of two natural numbers 'm' and 'n'.

Input Format:
Input consists of 2 natural numbers.

Output Format:
Refer sample input and output for formatting specifications.

Sample Input and Output:
Enter first natural number 'm' : 40
Enter second natural number 'n' : 130

LCM(35,120) = 260

16. Series Problem #1

Consider the following series:

1, 1, 2, 3, 4, 9, 8, 27, 16, 81, . . .

If you note carefully, you will find that even term represent one GP and odd term represent second GP.

Write a program to generate first 'n' terms of this series when common ratios of these two GPs are given as 'r1' and 'r2', respectively.

Input Format:
Input consists of three integers, 'n', 'r1' and 'r2', in order, on separate lines.

Output Format:
Display the first 'n' terms of the series.

Sample Input:
12
2
3
Sample Output:
1 1 2 3 4 9 8 27 16 81 32 243

17. Series Problem #2

Consider the following series:

1, 1, 2, 5, 4, 9, 8, 13, 16, 17, . . .

If you note carefully, you will find that odd term represent GP with common ratio of 2, and even term represent AP with a common difference of 4.

Write a program to generate first 'n' terms of this series when 'r' as common ratio of GP, and 'd' is the common difference of AP.

Input Format:
Input consists of three integers, n, r and d, in order, on separate lines.

Output Format:
Display the first 'n' terms of the series.

Sample Input:
12
2
4
Sample Output:
1 1 2 5 4 9 8 13 16 17 32 21

18. Series Problem #3

Consider the following series:

1, 1, 1, 5, 2, 9, 3, 13, 5, 17, 8, 21, 13,. . .

If you note carefully, you will find that odd term represent terms of Fibonacci sequence, and even term represent AP with a common difference of 4.

Write a program to generate first 'n' terms of this series with 'd' as the common difference of AP.

Input Format:
Input consists of two integers, 'n' and 'd' on separate lines.

Output Format:
Display the first 'n' terms of the series.

Sample Input:
10
4
Sample Output:
1 1 1 5 2 9 3 13 5 17

19. Series Problem #4

Consider the following series:

20, 1, 16, 1, 12, 2, 8, 3, 4, 5, 0, 8, -4, 13, -8,. . .

If you note carefully, you will find that the odd terms represent AP with a common difference of -4, and even terms represent term of Fibonacci sequence.

Write a program to generate first 'n' terms of this series with 'd' as the common difference of AP.

Input Format:
Input consists of two integers, 'n' and 'd' on separate lines.

Output Format:
Display the first 'n' terms of the series.

Sample Input:
10
-4
Sample Output:
20, 1, 16, 1, 12, 2, 8, 3, 4, 5.

20. Decimal to Binary Conversion

Write a program to perform decimal to binary conversion.

Input Format:
Input consists of a single decimal number.

Output Format:
Print the equivalent binary number.

Sample Input and Output:
Enter the decimal number: 10
Binary equivalent of decimal number 10 is 1010.

21. Binary to decimal Conversion

Write a program to perform binary to decimal conversion.

Input Format:
Input consists of a single binary number.

Output Format:
Print the equivalent decimal number.

Sample Input and Output:
Enter the binary number: 1010
Decimal equivalent of binary number 1010 is 10.

22. Sorting

Write a program to sort elements of an array named 'a' of size 'n'. Elements of array are of type 'int'.

Input Format:
Input consists of 'n+1' integers. The first integer corresponds to 'n'. The next 'n' integers correspond to the elements in the array.

Output Format:
Display the elements of the sorted array.

Sample Input:
5
3
2
6
8
1
Sample Output:
1 2 3 6 8

23. First 3-Largest Numbers

Write a program to find the first 3-largest numbers from an array named 'a' of size 'n', without sorting the array. The elements of the array are of type 'int'.

Input Format:
Input consists of 'n+1' integers. The first integer corresponds to 'n'. The next 'n' integers correspond to the elements in the array.

Output Format:
Display the first 3-largest numbers in order.
Sample Input:
10
35 20 61 80 19 40 100 75 57 60

Sample Output:
100
80
75

24. Leaders

An element is a leader if it is greater than all the elements to its right side, i.e., following it. And the rightmost element is always a leader.

For example in the array {16, 17, 4, 3, 5, 2}, leaders are 17, 5 and 2.

Write a program to print all the leaders in the array.

Input Format:
The first line of the input consists of an integer 'n' that corresponds to the number of elements in the input array.
The second line consists of elements of array separated by space.

Output Format:
List of numbers that are leaders, one single line, separated by space.

Sample Input 1:
6
16 17 4 3 5 2
Sample Output 1:
17 5 2

Sample Input 2:
6
16 10 4 3 18 19
Sample Output 2:
19

25. Majority Element

A majority element in an array of size 'n' is an element that appears more than 'n/2' times.

1. If there exists a majority element, then display that element.
2. If such element doesn't exist, then display "No Majority Element exists".

Write a program that prints the majority element in an array.
Input Format:
The first line of the input consists of an integer 'n' that corresponds to the number of elements in the input array.
The second line consists of elements of array separated by space.

Output Format:
Refer sample output formatting specifications.

Sample Input 1:
9
3 3 4 2 4 4 2 4 4
Sample Output 1:
4

Sample Input 2:
8
3 3 4 2 5 4 2 4 6
Sample Output 2:
No Majority Element exists.

26. Sorting a String

Write a program to sort a string in alphabetical (dictionary) order.

Input Format:
Input consists of a string. Assume that the input string consists of only letters and the maximum length of the string is 50.

Output Format:
A string whose letters are sorted in alphabetical order.

Sample Input:
Anitha

Sample Output:
Aahint

27. Duplicate Elements

Given an array of 'n<=50' elements which contains elements from 1 to n, generated randomly, with any of these numbers appearing any number of times.

Write a program to find these repeating numbers.

Example:
Let A = {1, 2, 3, 1, 3, 6, 6, 3}
Duplicate elements = 1 3 6

Input Format:
A single number 'n' that corresponds to the number of elements in the array.

Output Format:
Refer sample output formatting specifications.

Sample Input 1:
8

Let us assume that following are the elements generated randomly:

3 1 5 3 2 1 2 6

Sample Output 1:
3 1 2

Sample Input 2:
5

Let us assume that following are the elements generated randomly:

3 1 5 4 2

Sample Output 2:
No duplicates

28. Distinct Elements

Given an array of 'n<=50' elements which contains elements from 1 to n, generated randomly, with any of these numbers appearing any number of times.

Write a program to find these repeating numbers.

Example:
Let A = {1, 2, 3, 1, 3, 6, 6, 5}
Distinct elements = 2 5

Input Format:
A single number 'n' that corresponds to the number of elements in the array.

Output Format:
Refer sample output formatting specifications.

Sample Input 1:
8

Let us assume that following are the elements generated randomly:

1, 2, 3, 1, 3, 6, 6, 5

Sample Output 1:
2 5

Sample Input 2:
6

Let us assume that following are the elements generated randomly:

3 2 5 3 2 5

Sample Output 2:
No distinct element.

29. To Remove Duplicate Elements

Given an array of 'n<=50' elements. Write a program to remove duplicate elements from the array.

Input Format:
The first line of the input consists of an integer 'n' that corresponds to the number of elements in the input array.
The next 'n' lines in the input correspond to the elements in the array.

Output Format:
Refer sample output formatting specifications.

Sample Input:
10
10
22
31
10
31
16
26
45
22
31

Sample Output 1:
10
22
31
16
26
45

30. Missing Number

Write a program to find the missing integer among the array of 'n<=9' integers that are in the range 1 to (n+1).

Input Format:
The first line of the input consists of an integer 'n' that corresponds to the number of elements in the input array.

The second line consists of 'n' integers in the range 1 to (n+1), separated by space.

Output Format:
List of missing number(s).

Sample Input 1:
5
1 5 3 2 6

Sample Output 1:
4

31. Sum

Given an unsorted array 'a' of size 'n' of non-negative integers, find a continuous sub-array which adds to a given number sum.

Input Format:
The first line contains an integer 'n' denoting the size of the array.
The second line contains integers denoting the elements
of the array separated by space.
The last line contains an integer, denoting the sum.

Output Format:
The output line contains integers denoting the indexes of the sub-array.

Sample Input 1:
6
1 4 3 0 5 10
7

Sample Output 1:
Sum found between indexes 1 and 4

Sample Input 2:
7
1 4 3 0 5 10
9

Sample Output 2:
No sub-array found.

32. Equilibrium

Equilibrium position in an array is a position such that the sum of elements before of it is equal to the sum of elements after it.

Given an array 'a' of size 'n' of non-negative integers, find the index in the array where equilibrium occurs first in the array.

Input Format:
The first line contains an integer 'n' denoting the size of the array.
The second line contains integers denoting the elements of the array separated by space.

Output Format:
The output line contains integers denoting the equilibrium position or -1 if no equilibrium exists.

Sample Input 1:
7
-7 1 5 2 -4 2 1

Sample Output 1:
3

Sample Input 2:
7
1 4 3 0 5 10

Sample Output 2:
-1

33. Reverse Sub-array

Given an array 'a' of size 'n' of non-negative integers. Write a program to reverse every sub-array of size 'k', where 'n' is multiple of 'k'.

Input Format:
The first line contains an integer 'n' denoting the size of the array.
The second line contains integers denoting the elements of the array separated by space.
The last line contains an integer 'k' denoting the size of the sub-array.

Output Format:
The output line contains integers denoting the elements of the array, with elements of each sub-array reversed..

Sample Input:
6
-7 1 5 6 -4 2

Sample Output:
5 1 -7 2 -4 6

34. ASCII to Integer Conversion

Your task is to implement the function atoi. The function takes a string 'str' denoting a decimal number string, converts it to integer and returns it.

Input Format:
A numeric string.

Output Format:
Integer value

Sample Input:
"8671"

Sample Output:
8671

35. Triplets

Given an array 'a' of size 'n'. Write a program to find all the triplets such that sum of any two elements is equal to the third element.

Input Format:
The first line contains single integers 'n' denoting the size of the array.
The next line contains the elements of the array, separated by space.

Output Format:
The output line contains all those triplets whose sum of any two elements is equal to the third element.

Sample Input:
8
7 2 5 6 -4 6 3 9 8

Sample Output:
(7,2,5) (6,3,9)

36. Counting Bits

Write a program that counts the bits that are 1 in the binary representation of natural number 'n'.

Input Format:
Input consists of a single natural number 'n'.

Constraint:
20 <= N <=1000

Output Format:
Output consists of the counts of 1s in the binary representation of 'n'.

Sample Input 1:
127 (01111111)

Sample Output 1:
Number of 1-bits in 127 = 7

Sample Input 2:
32 (100000)

Sample Output 2:
Number of 1-bits in 32 = 1

37. Left Rotation

A *left rotation* operation on an array of size n shifts each of the array's elements 1 unit to the left. Given an integer, d, rotate the array that many steps left. **Example,**
d = 2
arr = [1,2,3,4,5]
After 2 rotations, arr' = [3,4,5,1,2].
Input Format:
The first line contains two space-separated integers that denote the number of integers, and the number of left rotations to perform.
The second line contains space-separated integers that represent the elements of the array.

Output Format:
List of elements of left rotated array.

Constraints
- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

Sample Input:
5 4
1 2 3 4 5

Sample Output:
5 1 2 3 4

38. Magic Triplets

Given an array 'a' of 'n' integers, a triplet (a[i], a[j], a[k]) is called Magic Triplet if a[i] < a[j] < a[k] and i < j < k. The task is to count number of magic triplets in a given array.

Input Format:
The first line contains an integer 'n' denoting the size of array.

The second line contains 'n' space-separated integers denoting the elements of the array.

Output Format:
Single number representing the number of triplets.

Constraints:
1 <= n <= 10^3
1 <= a[i] <= 5000

Sample Input 1:
3
3 2 1
Sample Output 1:
0

Sample Input 2:
4
1 2 3 4
Sample Output 2:
2

39. Numbers with alternate bits

Input Format:
Input consists of a single natural number 'n'.

Output Format:
Refer sample output formatting specifications.

Sample Input 1:
5 (101)

Sample Output 1:
Number has alternate pattern of 0s and 1s.

Sample Input 2:
15 (1111)

Sample Output 2:
Number doesn't have alternate pattern of 0s and 1s.

40. EVEN or ODD

Write a program to check whether the given natural number is EVEN or ODD.

Input Format:
Input consists of a single natural number 'n'.

Constraints:
1. Modulus operation by 2 as well 10 is not
2. Use of *if* statement is not permitted.

Output Format:
Refer sample output formatting specifications.

Sample Input 1:
Enter natural number : 35

Sample Output 1:
35 is odd number.

Sample Input 2:
Enter natural number : 120
.
Sample Output 2:
120 is even number.

41. Swapping

Write a program to swap two natural numbers.

Input Format:
Input consists of a two natural number *a* & *b* separated by space.

Constraints:
1. Use of third variable is not allowed
2. Only single statement is permitted to facilitate the swapping.
3. 0 < *a*, *b* ≤ 10^7

Output Format:
Refer sample output formatting specifications.

Sample Input:
Enter two natural number a & b : 20 25

Sample Output:
After swapping
a = 25, b = 20