JSpiders
Training & Development Center
JNTU HYDERABAD
9980300600

Java

Topic wise Practice Questions

Java

JSpiders PySpiders Training and Development Center

JNTU HYDERABAD

INDIA USA UK IRELAND

JSpiders

JNTU Hyderabad

Ph:-9980300600

JAVA QUESTION BANK – Version 2

Programming

Chapter 1

- **Core Level Programs**
  - Number Programs
  - Pattern Programs
  - Array Programs
  - String Programs

Chapter 2

- **Medium Level Programs**
  - Number Programs
  - Pattern Programs
  - Array Programs
  - String Programs

Chapter 3

- **Advanced Level Programs**
  - Number Programs
  - Pattern Programs
  - Array Programs
  - String Programs

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Chapter 1

Core Level Programs

Number Programs

1. WAPT check whether the given number is even or odd
2. WAPTP sum of the numbers between m to n
3. WAPTP product of the numbers between m to n
4. WAPT count the numbers from m to n
5. WAPT swap two numbers
6. Using third variable
7. Without using third variable
8. WAPTP square of a given number
9. WAPTP cube of a given number
10. WAPTP factorial value of a number
11. WAPTP factors of the number
12. WAPTP Fibonacci series
13. WAPTP the exponential value for a given base and power (x ^ n)
14. WAPT extract digits in reverse order
15. Print
16. Count
17. Sum
18. Product
19. WAPT Reverse a Number.

Pattern Programs

Solid square of `*` (5x5)

```
*****
*****
*****
*****
*****
```

Hollow square of `*` (5x5)

```
*****
*   *
*   *
*   *
*****
```

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Square with diagonal elements as `*`, rest as space

```
*   *
 * *
  *
 * *
*   *
```

Right-angled triangle using `*`

```
*
**
***
****
*****
```

Inverted right-angled triangle using `*`

```
*****
****
***
**
*
```

Full pyramid using `*`

```
    *
   ***
  *****
 *******
*********
```

Inverted full pyramid using `*`

```
*********
 *******
  *****
   ***
    *
```

Diamond Pattern (Size = 5)

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Hollow Diamond Pattern

```
  *
 * *
*   *
*    *
*     *
*    *
*   *
 * *
  *
```

Array Programs

1. WAPT Print all elements in an array.
2. WAPT to count the No. of elements in an array.
3. WAPT Print the elements of an array in reverse order.
4. WAPT Print all elements at even indices in an array.
5. WAPT Calculate the sum of all elements in an array.
6. WAPT Print all duplicate elements in an array.
7. WAPT Sort an array without using predefined methods.
8. WAPT Merge two arrays.
9. WAPT Print the largest element in a given array.
10. WAPT Print the smallest element in a given array.
11. WAPT Find the frequency of each element in an array.
12. WAPT to copy all elements from one array to another array.

String Programs

1. WAPT extract all the characters present in the given String.
2. WAPT to print
   - Only Alphabets
   - Only Digits
   - Only Special Characters
3. WAPT Reverse a given String
4. WAPT extract all the duplicate characters from a given string.
5. WAPT extract all the words present in the given String.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Chapter 2

Medium Level Programs

Number Programs

13. WAP to Print the multiplication table of a number
14. WAP to check whether the given number is prime or not.
15. WAP to print the sum of prime numbers from m to n
16. WAP to print the product of prime numbers from m to n
17. WAP to print the sum of the factors of a number.
18. WAP to print the product of the factors of a number.
19. WAP to print sum of even and odd factors in a given number.
20. WAP to print product of even and odd factors in a given number
21. WAP to count the digits in a given number.
22. WAP to print even digits in a given number.
23. WAP to print odd digits in a given number
24. WAP to print sum of digits in a given number
25. WAP to print product of digits in a given number
26. WAP to print sum of even digits in a given number
27. WAP to print sum of prime digits in a given number
28. WAP to print product of prime digits in a given number.
29. WAP to print the factorial of each digit in a given number.
30. WAP to print the sum of factorial of each digit in a given number.
31. WAP to print the factorial of even digits in a given number
32. WAP to print the factorial of odd digits in a given number.
33. WAP to print the power of each digit to the count number of digits in the given number.
34. WAP to print the sum of power of each digit to the count number of digits in the given number.
35. WAP to print the sum of power of even digit to the count number of digits in the given number.
36. WAP to print the sum of power of odd digit to the count number of digits in the given number.
37. WAP to Print the nth prime number.
38. WAP to Print the next prime number for a given number.
39. WAP to Print the nth Fibonacci number
40. WAP to Check if a number is a Fibonacci number.
41. WAP to Find the largest digit in a given number.
42. WAP to Find the smallest digit in a given number.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

43. WAP to Find the nth Largest digit in a given number.
44. WAP to Find the nth smallest digit in a given number.
45. WAP to Find the sum of smallest digit and largest digit in a given number.
46. WAP to Find the sum of nth smallest digit and nth largest digit in a given number.
47. WAP to Find the binary representation of a number.
48. WAP to Find the decimal representation of a binary number.
49. WAP to Find the GCD or HCF of two numbers.
50. WAP to Find the LCM of two numbers.

Pattern Programs

Square Number Pattern

Question: Print a square where each row contains the same number from 1 to n.

Input: n = 4

Output:

```
11111
22222
33333
44444
```

Question: Print a square where each row contains numbers from 1 to `n`.

Input: n = 4

Output:

```
12345
12345
12345
12345
```


Hollow Number Square

Question: Print a hollow square with numbers at the border and space inside.

Input: n = 5

Output:

```
12345
1   5
1   5
1   5
12345
```

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Alternating 1s and 0s

Question: Print a square pattern with alternating 1s and 0s like a chessboard.

Input: n = 5

Output:

```
10101
01010
10101
01010
10101
```

Sum of Indices Pattern

Question: Print a square where each cell contains the sum of row and column indices (starting from 0).

Input: n = 4

Output:

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```

Number Triangle

Question: Print a triangle with increasing numbers in each row.

Input: n = 5

Output:

```
1
12
123
1234
12345
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Inverted Number Triangle

Question: Print a triangle with decreasing numbers in each row.

Input: n = 5

Output:

```
12345
1234
123
12
1
```

Floyd's Triangle

Question: Print Floyd's triangle with continuous numbers.

Input: n = 5

Output:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Number Pyramid

Question: Print a centred pyramid of increasing numbers.

Input: n = 5

Output:

```
    1
   121
  12321
 1234321
123454321
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Inverted Number Pyramid

Question: Print an inverted number pyramid with decreasing width.

Input: n = 5

Output:

```
123454321
 1234321
  12321
   121
    1
```

Alphabet Pyramid

Question: Print a centred pyramid with alphabets increasing in each row.

Input: n = 5

Output:

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

Inverted Alphabet Pyramid

Question: Print an inverted pyramid with alphabets decreasing in each row.

Input: n = 5

Output:

```
ABCDEDCBA
 ABCDCBA
  ABCBA
   ABA
    A
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Floyd's Triangle in Pyramid Form

Question: Print Floyd's triangle shaped as a pyramid.

Input: n = 5

Output:

```
   1
  2 3
 4 5 6
7 8 9 10
11 12 13 14 15
```

Odd Number Pyramid

Question: Print a pyramid where each row contains increasing odd numbers.

Input: n = 5

Output:

```
   1
  3 5
 7 9 11
13 15 17 19
21 23 25 27 29
```

Even Number Pyramid

Question: Print a pyramid where each row contains increasing even numbers.

Input: n = 5

Output:

```
   2
  4 6
 8 10 12
14 16 18 20
22 24 26 28 30
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Number Diamond

- Print a diamond pattern with fixed number in each row.

  Input: n = 3

  Output:

  ```
   1
  222
  33333
  222
   1
  ```

- Print a diamond pattern using only numbers increasing by 1 on each line.

  Input: n = 3

  Output:

  ```
    1
   123
  12345
   123
    1
  ```

Character Diamond

- Print a diamond pattern using alphabets increasing row-wise (A to E).

  Input: n = 3

  Output:

  ```
   A
  BBB
  CCCCC
  BBB
   A
  ```


- Print a diamond pattern where each row contains consecutive alphabets.

  Input: n = 3

  Output:

  ```
    A
   ABC
  ABCDE
   ABC
    A
  ```

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Array Programs

1. Calculate the sum of all even elements in an array.
2. Calculate the sum of all odd elements in an array.
3. Calculate the sum of the first and last elements in an array.
4. Calculate the sum of the last two elements in an array.
5. Calculate the sum of all prime numbers in a given array.
6. Calculate the multiplication of all elements in an array.
7. Calculate the multiplication of all even elements in an array.
8. Calculate the multiplication of all prime numbers in a given array.
9. Calculate the average value of all elements in a given array.
10. Check if the multiplication of the last two elements in an array is even.
11. Write a Java program to find the frequency of each element in an array.
12. Remove the most repeated elements from an array.
13. Remove the most repeated even elements from an array.
14. Write a Java program to remove duplicates from a sorted array.
15. Print duplicate elements at even indices in an array.
16. Merge two arrays and find duplicate elements in the merged array.
17. Calculate the sum of duplicate elements in a given array.
18. Sort only the positive elements in a given array.
19. Sort only the even elements in a given array.
20. Sort only the prime elements in a given array.
21. Merge two arrays and sort the merged elements.
22. Sort only the first two and last two elements in a given array.
23. Print the second largest element in an array.
24. Print the second smallest element in an array.
25. Print the nth largest element in a given array.
26. Print the nth smallest element in a given array.
27. Print the largest even element in a given array.
28. Print the smallest odd element in a given array.
29. Print the largest prime number in a given array.
30. Calculate the sum of the largest and smallest elements in an array.
31. Calculate the multiplication of the largest and smallest elements in an array.
32. Calculate the average of the largest and smallest elements in an array.
33. Calculate the difference between the largest and smallest elements in an array.
34. Merge only the even elements from two arrays into a single array and print it in reverse order.
35. Sort two arrays in ascending order, merge them, and print only the even numbers.
36. Sort the prime numbers from two arrays and merge them into a single array.
37. Merge the largest and smallest elements from two arrays into a single array.
38. Print all prime numbers in an array.
39. Calculate the sum of the last four even numbers in an array.
40. Write a Java program to rotate an array to the left by a given number of steps.
41. Write a Java program to rotate an array to the right by a given number of steps.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

42. Write a Java program to find the first non-repeating element in an array.
43. Write a Java program to check if an array is a palindrome.
44. Write a Java program to remove an element from an array at a specific position.
45. Write a Java program to check if two arrays are equal or not.
46. Count the number of elements in an array without using the .length property.
47. Write a Java program to find the maximum product of two integers in an array.
48. Write a Java program to calculate the prefix sum array.
49. Write a Java program to find pairs in an array with a given sum.

String Programs

1. Reverse a string while maintaining the position of spaces.
2. Check if a string is a palindrome.
3. Count the frequency of each character in a string.
4. Find the first non-repeated character in a string.
5. Find the second most frequent character in a string.
6. Find the first repeating character in a string.
7. Reverse only the digits present in a string.
8. Replace duplicate characters with '*'.
9. Remove a specific character from a string.
10. Check if a string is made up of unique characters.
11. Find the longest palindrome in a string.
12. Find the smallest palindrome in a string.
13. Find the smallest and largest words in a string.
14. Find the frequency of each word in a string.
15. Reverse each word in a string individually.
16. Replace all spaces in a string with '%20'.
17. Check if two strings are rotations of each other.
18. Implement your own substring() method.
19. Implement your own split() method for a string.
20. Implement your own indexOf() method for strings.
21. Implement your own trim() method for strings.
22. Implement your own toLowerCase() and toUpperCase() methods.
23. Convert a string to its ASCII representation.
24. Convert a string to its equivalent integer representation.
25. Replace all vowels in a string with a given character.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Chapter 3

Advance Level Programs

Number Programs

1. Twisted Prime Number
- A **Twisted Prime Number** is a prime number that remains prime when its digits are reversed.
2. Mega Prime Number
- A **Mega Prime Number** is a prime number where **all its digits are also prime numbers**.
3. Palindrome Number
- A **Palindrome Number** is a number that remains the same when its digits are reversed
4. SPY Number
- A **SPY Number** is a number where the **sum of its digits is equal to the product of its digits.**
5. Perfect Number
- A **Perfect Number** is a number that is **equal to the sum of its factors (excluding the number itself).**
6. Strong Number
- A **Strong Number** is a number whose **sum of the factorial of its digits is equal to the original number.**
7. Neon Number
- A **Neon Number** is a number where the **sum of the digits of its square is equal to the original number.**
8. Armstrong Number
- An **Armstrong Number** is a number in which the **sum of its digits raised to the power of the total number of digits is equal to the number itself.**
9. Sunny Number
- A **Sunny Number** is a number where **the number + 1 is a perfect square.**
10. Automorphic Number
- An **Automorphic Number** is a number whose **square ends with the same digits as the number itself.**
11. Magic Number
- A **Magic Number** is a number in which the **recursive sum of digits (until a single digit) is 1.**

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

12.Tech Number

- A **Tech Number** is a number with **even number of digits**, which when split into two equal halves and added together, the **square of the sum is equal to the original number.**

13.Harshad Number

- A **Harshad Number** (also called Niven Number) is a number which is **completely divisible by the sum of its digits.**

Level – I

1. WAP to check whether the given number is Twisted Prime Number or not.
2. WAP to check whether the given number is a Mega Prime Number or not.
3. WAP to check whether the given number is a Palindrome Number or not
4. WAP to check whether the given number is a SPY Number or not.
5. WAP to check whether the given number is a Perfect Number or not.
6. WAP to check whether the given number is a Strong Number or not.
7. WAP to check whether the given number is a Neon Number or not.
8. WAP to check whether the given number is an Armstrong Number or not.
9. WAP to check whether the given number is a Sunny Number or not.
10. WAP to check whether the given number is an Automorphic Number or not.
11. WAP to check whether the given number is a Magic Number or not.
12. WAP to check whether the given number is a Tech Number or not.
13. WAP to check whether the given number is a Harshad (Niven) Number or not.

Level – II

1. WAP to print the Twisted Prime numbers present in the range of m to n.
2. WAP to print the Mega Prime numbers present in the range of m to n.
3. WAP to print the Palindrome numbers present in the range of m to n.
4. WAP to print the SPY numbers present in the range of m to n.
5. WAP to print the Perfect numbers present in the range of m to n.
6. WAP to print the Strong numbers present in the range of m to n.
7. WAP to print the Neon numbers present in the range of m to n.
8. WAP to print the Armstrong numbers present in the range of m to n.
9. WAP to print the Sunny numbers present in the range of m to n.
10. WAP to print the Automorphic numbers present in the range of m to n.
11. WAP to print the Magic numbers present in the range of m to n.
12. WAP to print the Tech numbers present in the range of m to n.
13. WAP to print the Harshad (Niven) numbers present in the range of m to n.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Level – III

1. WAP to print the nth Prime number.
2. WAP to print the nth Twisted Prime number.
3. WAP to print the nth Mega Prime number.
4. WAP to print the nth Palindrome number.
5. WAP to print the nth Sunny number.
6. WAP to print the nth Automorphic number.
7. WAP to print the nth Magic number.
8. WAP to print the nth Harshad (Niven) number.

Miscellaneous Number Series

1. Print the Lucas series up to n terms
2. Print the factorial series up to n terms.
3. Generate the Collatz sequence for a number.
4. Find the GCD or HCF of three numbers.
5. Generate the first n terms of a Tribonacci sequence.

Pattern Programs

1. Zigzag Number Pattern in Square

Input: 4
Output:
```
 1  2  3  4
 8  7  6  5
 9 10 11 12
16 15 14 13
```

2. Pascal's Triangle (Number Pattern)

Input: 5
Output:
```
     1
    1  1
   1  2  1
  1  3  3  1
 1  4  6  4  1
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

3. Palindromic Number Triangle

Input: 5
Output:

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

4. Alphabet Palindromic Pyramid

Sample Input: 5

```
        A
      A B A
    A B C B A
  A B C D C B A
A B C D E D C B A
```

5. Number Mirror Pyramid

Sample Input: 5

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

6. Alternating Character Pyramid

Sample Input: 5

```
      A
    1 A 1
  A 1 A 1 A
1 A 1 A 1 A 1
```

7. Hollow Number Pyramid

Sample Input: 5

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


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

8. Zig-Zag Alphabet Pyramid

Sample Input: 5

```
A
 B C
  D E F
   G H I J
    K L M N O
```

9. Alphabet Full Diamond

Sample Input: 5

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
 ABCDCBA
  ABCBA
   ABA
    A
```

10. Hollow Diamond with Alphabets

Sample Input: 5

```
    A
   B B
  C   C
 D     D
E       E
 D     D
  C   C
   B B
    A
```


11. Double-Sided Number Diamond

Sample Input: 5

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

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

12. Number Hourglass

Sample Input: 5

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

13. Alphabet Hourglass

Sample Input: 5

```
E D C B A B C D E
  D C B A B C D
    C B A B C
      B A B
        A
      B A B
    C B A B C
  D C B A B C D
E D C B A B C D E
```

14. Hollow Hourglass with Alphabets

Sample Input: 5

```
A B C D E D C B A
  B       B
    C   C
      D D
        E
      D D
    C   C
  B       B
A B C D E D C B A
```


Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

Array Programs

Level-I

1. Check if the sum of all array elements is an Armstrong number.
2. Check if the product of all array elements is a palindrome.
3. Check if the sum of all even elements is a strong number.
4. Check if the sum of all odd elements is an Armstrong number.

Level-II

1. Write a Java Program for MergeSort

Problem:
Implement a Java program for MergeSort algorithm to sort an array in ascending order.
Example Input:
arr = [12, 11, 13, 5, 6, 7]
Example Output:
arr = [5, 6, 7, 11, 12, 13]

2. Write a Java Program for QuickSort

Problem:
Implement a Java program for QuickSort algorithm to sort an array in ascending order.
Example Input:
arr = [10, 7, 8, 9, 1, 5]
Example Output:
arr = [1, 5, 7, 8, 9, 10]

3. Replace every element with the next greatest element (from the right side) in a given array, replacing the last element with -1

Problem:
Replace every element of the array with the next greatest element from the right side. Replace the last element with -1.
Example Input:
arr = [16, 17, 4, 3, 5, 2]
Example Output:
arr = [17, 17, 5, 5, 5, -1]

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

4. Rearrange an array with alternate high and low elements

Problem:
Rearrange the elements of the array in a way such that the array alternates between high and low elements.
Example Input:
arr = [1, 2, 3, 4, 5, 6]
Example Output:
arr = [1, 6, 2, 5, 3, 4]

5. Find the missing element in a given array in the series of 1 to n

Problem:
Given an array with integers from 1 to n with one element missing, find the missing element.
Example Input:
arr = [1, 2, 4, 6, 3, 7, 8]
Example Output:
Missing Element: 5

6. Find the Majority Element (Boyer-Moore Algorithm)

Problem:
Given an array of size n, find the element that appears more than n/2 times (majority element). You can assume that the majority element always exists in the array.
Example Input:
arr = [2, 2, 1, 1, 2, 2]
Example Output:
Majority Element: 2

7. Check If Array Can Be Made Strictly Increasing by Removing One Element

Problem:
Given an array, determine if it's possible to make the array strictly increasing by removing at most one element. Return true if the array can be made strictly increasing by removing one element, otherwise false.
Example Input:
arr = [10, 1, 2, 3, 4]
Example Output:
True (Remove 10)

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

8. Find the Peak Element

Problem:
Find an element which is strictly greater than its neighbors. The array should not be empty, and you are required to solve it in O(log n) time.
Example Input:
arr = [1, 3, 20, 4, 1]
Example Output:
20

String Programs

1. Remove Consecutive Duplicates

Write a program to remove all consecutive duplicate characters in a string.

Example:
Input: "aaabbccdaa"
Output: "abcda"

2. Implement strstr() Function

Implement your own version of the strstr() function to find the first occurrence of a substring.

Example:
Input: "hello", "ll"
Output: 2

3. Remove Characters from First String Present in Second

Given two strings, remove all characters from the first string that appear in the second.

Example:
Input: str1 = "computer", str2 = "cat"
Output: "ompuer"

4. Check if Two Strings Are Isomorphic

Determine if two strings follow the same character pattern.

Example:
Input: "egg", "add"
Output: true

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

5. Run-Length Encoding

Compress a string using run-length encoding.

Example:
Input: "aaabbc"
Output: "a3b2c1"

6. Capitalize First Letter of Each Word

Write a program to capitalize the first letter of each word in a string.

Example:
Input: "hello world"
Output: "Hello World"

7. Find All Substrings of a String

Generate and print all possible substrings of a given string.

Example:
Input: "abc"
Output: "a", "ab", "abc", "b", "bc", "c"

8. Check if One String is a Permutation of Another

Check whether one string is a permutation of another string.

Example:
Input: "abc", "bca"
Output: true

9. Check if a String is a Valid Palindrome Ignoring Non-Alphanumerics

Determine if a string is a palindrome, considering only alphanumeric characters and ignoring cases.

Example:
Input: "A man, a plan, a canal: Panama"
Output: true

10. Minimum Window Substring

Given two strings s and t, return the minimum window in s which contains all the characters of t.

Example:
Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

11. Check if a Pattern Matches a String (Word Pattern)

Given a pattern and a string, determine if the string follows the same pattern.

Example:
Input: pattern = "abba", str = "dog cat cat dog"
Output: true

12. Longest Substring Without Repeating Characters

Find the length of the longest substring without repeating characters.

Example:
Input: "abcabcbb"
Output: 3

13. Decode Encoded String (Nested Encoding)

Decode an encoded string where patterns are in the form k[encoded_string].

Example:
Input: "3[a2[c]]"
Output: "accaccacc"

14. Reverse a String Without Affecting Special Characters

Reverse the characters in a string, but keep the special characters in their original position.

Example:
Input: "a,b$c"
Output: "c,b$a"

15. Check if Two Strings Are One Edit Away

Check whether two strings are one edit (insert, remove, replace) away from each other.

Example:
Input: "pale", "ple"
Output: true

16. Longest Palindromic Subsequence

Find the longest subsequence in a string that is also a palindrome.

Example:
Input: "bbabcbcab"
Output: 7 ("babcbab")

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

17. Remove Duplicate Letters to Get Lexicographically Smallest Result

Remove duplicate letters so that every letter appears once and the result is the smallest in lexicographical order.

Example:
Input: "cbacdcbc"
Output: "acdb"

18. Minimum Insertions to Make a String Palindrome

Find the minimum number of insertions required to make a string a palindrome.

Example:
Input: "abcda"
Output: 2

19. Smallest Substring Containing All Characters of Another String

Find the smallest substring in a given string that contains all characters of another string.

Example:
Input: s = "this is a test string", t = "tist"
Output: "t stri"

20. Check If Two Strings Are Anagrams

Write a program to check if two strings are anagrams of each other.

Example:
Input: "listen", "silent"
Output: true

21. Group All Anagrams Together

Given an array of strings, group the anagrams together.

Example:
Input: ["eat", "tea", "tan", "ate", "nat", "bat"]
Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]

22. Check if Two Strings Are K-Anagrams

Two strings are k-anagrams if they can be made anagrams by changing at most k characters in one of the strings.

Example:
Input: str1 = "anagram", str2 = "grammar", k = 3
Output: true

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME

23. Find All Anagram Indices

Given a string and a pattern, return all starting indices of substrings in the string that are anagrams of the pattern.

Example:
Input: s = "cbaebabacd", p = "abc"
Output: [0, 6]

24. Check If One String is a Rotation of Another and an Anagram

Check whether one string is a rotation and also an anagram of another string.

Example:
Input: "abcde", "cdeab"
Output: true

Note:

- The listed programming questions are **important and useful for practice**, but they do **not cover the full scope** of what can be asked in coding interviews.
- Programming involves **a wide variety of problem types** beyond what's included here. Topics such as dynamic programming, bit manipulation, recursion, and others are **also crucial**.
- The problems listed are considered **foundational or frequently encountered** during coding practice. They help in building problem-solving skills, but **should be complemented** with real-world problem scenarios and mock interview preparation.
- here is **no predefined or guaranteed set** of questions that will be asked in interviews. Each company, interviewer, and job role may focus on different problem types and difficulty levels.

Note: 1. All programs should be implemented with Scanner class for input operations.
2. Avoid using inbuilt or predefined methods in the programs.
3. Each program should include user-defined methods.

HOME


“If you are perfect in all the above listed questions, you can confidently face any challenge that comes your way in an interview."

```
if(hardworking && patience)
{
 System.out.println("Depend on
JSPIDERS JNTU for Interviews");
}
```