PAQ’s Coding Automata AMCAT (Similar Questions You Can Expect With 1 easy and 1 hard difficulty)

1. Program to check if the given number is even or odd
2. Program to check if the given number is a Prime number
3. Program to check if the given number is Palindrome
4. Program to check if the given number is Perfect number
5. Program to check if the given number is Armstrong number
6. Find the factorial of a number
7. Program to check if the given number is Strong number
8. Program to count the digits of a number
9. Program to swap two numbers (All methods)
10. Program to print Fibonacci series
11. Find the largest of two numbers without conditional statement
12. Program to find the area of a Triangle when sides are given
13. Program to find the volume of a Sphere when radius is given
14. Program to check if the year is a Leap year
15. Program to convert Binary to Decimal and viceversa
16. Program to convert Binary to Octal
17. Program to find LCM of two numbers without recursion
18. Program to find GCD of two numbers without recursion
19. Program to print wedge, pyramid and diamond pattern
20. Find the smallest and Largest number in the array
21. Find the second largest number in the array
22. Program to remove duplicate elements in an array
23. Program to search for an element in an array using linear search
24. Program to sort an array in ascending order using Bubble Sort
25. Program to find the Largest Sum of Contiguous Subarray - Kadane's Algorithm
26. Find the frequency of numbers in an array
27. Program to separate Even and Odd numbers in an array
28. Check whether a given number is an **Armstrong number**.
29. Find the **frequency of each digit** in a given number.
30. Calculate the **HCF (Highest Common Factor)** of two given numbers.
31. Find the **largest number in an array** of integers.
32. Calculate the **LCM (Least Common Multiple)** of two given numbers.
33. Check whether a given number or string is a **palindrome**.

34. Check whether a given number is a **perfect number**.
35. Check whether a given number is a **prime number**.
36. Reverse a given **string**.
    Perform **addition of two matrices** of the same dimensions.

37. Check whether a given matrix is an **identical matrix**.

38. Print the elements of a matrix in **spiral order**.

39. Find the **transpose of a given matrix**
40. Print a **full pyramid pattern** using stars.
    Print a **half pyramid pattern** using stars.
    Print a **half pyramid pattern using numbers**.
    Print an **inverted half pyramid pattern** using stars.
41. Program to print all **distinct elements** of given input arrays. Also print the **total of the distinct elements**.

    Input:
    Arr1 = {1, 2, 3, 4, 5}
    Arr2 = {2, 6, 8, 10}

42. Find the **occurrence of a substring** in a parent string.

    Input:
    Parent String: aAbcDefabcAdf
    Substring: abc

43. Write a program to **swap two numbers without using a third variable**.

44. Write a program to find the **sum of digits** of a given number.
45. Print the following **pattern** if the input is **5**:

```
    1
    2*3
    4*5*6
    7*8*9*10
    7*8*9*10
    4*5*6
    2*3
   1
```

46.Print the pattern if **Input: N = 4 and S = 3**, where **S is the first number**.

```
3
44
555
6666
6666
555
44
3
```

47. Print the pattern if **input is 5**.

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

48. Mooshak the mouse has been placed in a **maze**. There is a huge chunk of **cheese** somewhere in the maze.
The maze is represented as a **two-dimensional array of integers**, where:

- 0 represents **walls**
- 1 represents **paths** where Mooshak can move
- 9 represents the **huge chunk of cheese**

Mooshak starts at the **top-left corner (0,0)**.

Problem Statement

Write a method **isPath** of class **MazePath** to determine whether Mooshak can reach the huge chunk of cheese.

- The input to isPath consists of a **two-dimensional array** representing the maze matrix.
- The method should return **1** if there is a path from Mooshak to the cheese.
- The method should return **0** if Mooshak **cannot** reach the cheese.
- Mooshak is **not allowed to leave the maze or climb on walls**.

Example

8 × 8 matrix maze where Mooshak can get the cheese:

```
1 0 1 1 1 0 0 1
1 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0
1 0 1 0 9 0 1 1
1 1 1 0 1 0 0 1
1 0 1 0 1 1 0 1
1 0 0 0 0 1 0 1
1 1 1 1 1 1 1 1
```

Test Cases

Test Case 1

Input:

```
[[0,0,0],
 [9,1,1],
 [0,1,1]]
```

Expected Return Value:

```
0
```

Test Case 2

Input:

```
[[1,1,1],
 [9,1,1],
 [0,1,0]]
```

Expected Return Value:

```
1
```

Explanation (Test Case 2)

The piece of cheese is placed at position **(1,0)** on the grid.

Mooshak can move:

- Directly from **(0,0)** → **(1,0)**
  OR
- From **(0,0)** → **(0,1)** → **(1,1)** → **(1,0)**

Thus, Mooshak can reach the cheese.

49 . Cell Competition Problem

There is a **colony of 8 cells** arranged in a straight line, where **each day every cell competes with its adjacent (neighbor) cells**.

Rules

- Each day, for every cell:
    - If **both neighbors are active** or **both neighbors are inactive**, the cell becomes **inactive** the next day.
    - Otherwise, the cell becomes **active** the next day.

Assumptions

- The two end cells have **only one adjacent cell**, so the **missing neighbor** is assumed to be **always inactive (0)**.
- Even after updating the cell state, **its previous state** should be considered while updating other cells.
- The cell information for **all cells is updated simultaneously**.

Function Description

Write a function **cellCompete** which:

- Takes an **8-element integer array** `cells` representing the current state of the 8 cells.
- Takes an integer `days` representing the number of days to simulate.
- A value of **1** represents an **active** cell.
- A value of **0** represents an **inactive** cell.

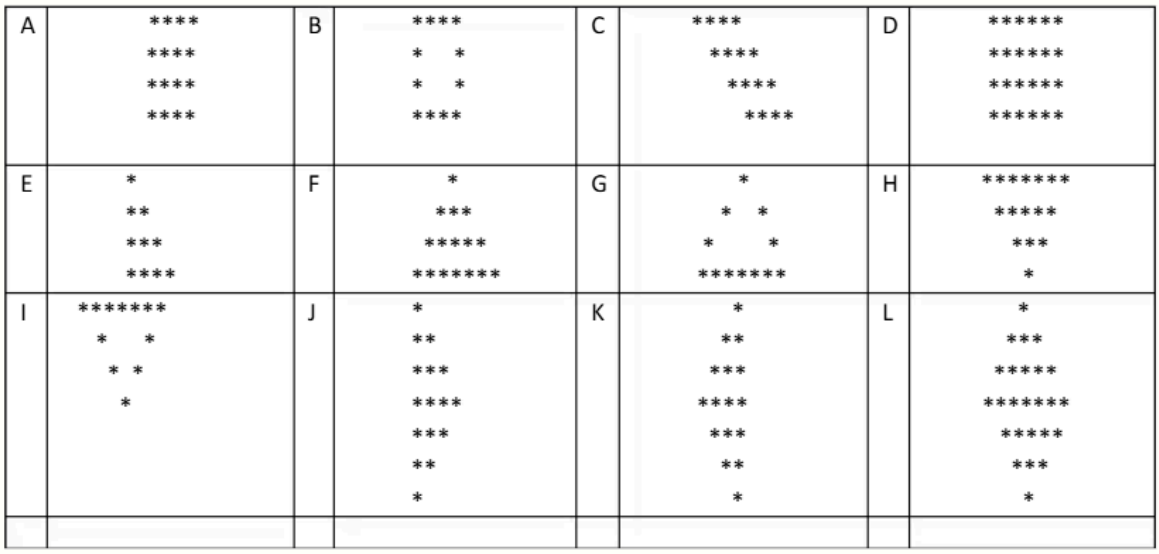
Input

```
cells = [1, 0, 0, 0, 0, 1, 0, 0]
days = 1
```

Expected Return Value

```
[0, 1, 0, 0, 1, 0, 1, 0]
```

50.Generate below patterns using code . Write full program for it



A	**** **** **** ****	B	**** * * * * ****	C	**** **** **** ****	D	****** ****** ****** ******
E	* ** *** ****	F	* *** ***** *******	G	* * * * * *******	H	******* ***** *** *
I	******* * * * * *	J	* ** *** **** *** ** *	K	* ** *** **** *** ** *	L	* *** ***** ******* ***** *** *