"A Puzzle a day keeps the placement fear away" **-Mayank Hanwat**

Tasks outline :

DAY 1 : Number of wins for each player in a series of Rock-Paper-Scissor game.

DAY 2 : Check if the given chessboard is valid or not.

DAY 3 : Find safe cells in a matrix.

DAY 4 : Find the non decreasing order array from given array.

DAY 5 : Count number of trailing zeros in (1^1)(2^2)*(3^3)*(4^4)*.....*

DAY 6 : Partition negative and positive without comparison with 0.

DAY 7 : Time taken by two persons to meet on a circular Track.

GeeksforGeeks

DAY 1

Number of wins for each player in a series of Rock-Paper-Scissor game

Two players are playing a series of games of Rock–paper–scissors. There are a total of **K** games played. Player 1 has a sequence of moves denoted by string **A** and similarly player 2 has string **B**. If any player reaches the end of their string, they move back to the start of the string. The task is to count the number of games won by each of the player when exactly **K** games are being played.

Examples:

Input: *k = 4, a = "SR", b = "R"*
Output: *0 2*
Game 1: Player1 = S, Player2 = R, Winner = Player2
Game 2: Player1 = R, Player2 = R, Winner = Draw
Game 3: Player1 = S, Player2 = R, Winner = Player2
Game 4: Player1 = R, Player2 = R, Winner = Draw
Input: *k = 3, a = "S", b = "SSS"*
Output: *0 0*
All the games are draws.

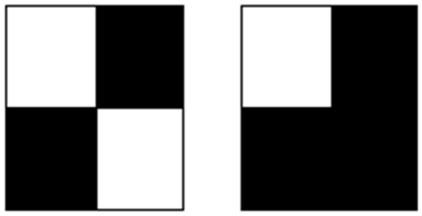
Approach: Let length of string **a** be **n** and length of string **b** be **m**. The observation here is that the games would repeat after **n * m** moves. So, we can simulate the process for **n * m** games and then count the number of times it gets repeated. For the remaining games, we can again simulate the process since it would be now smaller than **n * m**. For example, in the first example above, **n = 2** and **m = 1**. So, the games will repeat after every **n * m = 2 * 1 = 2** moves i.e. **(Player2, Draw)**, **(Player2, Draw)**, ….., **(Player2, Draw)**.

GeeksforGeeks

DAY 2

Check if the given chessboard is valid or not

Given a **N X N** chessboard. The task is to check if the given chessboard is valid or not. A chess board is considered valid if every 2 adjacent cells are painted with different color. Two cells are considered adjacent if they share a boundary.



First chessboard is valid whereas second is invalid.

Examples:

```
Input : N = 2
C = {
      { 1, 0 },
      { 0, 1 }
    }
Output : Yes


Input : N = 2
C = {
      { 0, 0 },
      { 0, 0 }
    }
Output : No
```

Approach: Observe, on a chess board, every adjacent pair of cells is painted in different color.
So, for each cell (i, j), check whether the adjacent cells i.e
(i + 1, j), (i -1, j), (i, j + 1), (i, j – 1) is painted with different color than (i, j) or not.

GeeksforGeeks

DAY 3

Find safe cells in a matrix

Given a matrix **mat [][]** containing the characters **Z**, **P** and ***** where **Z** is a zombie, **P** is a plant and ***** is a bare land. Given that a zombie can attack a plant if the plant is adjacent to the zombie (total 8 adjacent cells are possible). The task is to print the number of plants that are safe from the zombies.

Examples:

```
Input:
mat[] = { "**P*",
          "*Z**",
          "*Z**",
          "***P" }
Output: 1

Input:
mat[] = { "**P*P",
          "*Z**",
          "*P**",
          "***P" }
Output: 2
```

Approach: Traverse the matrix element by element, if the current element is a plant i.e. **mat [i][j] = 'P'** then check if the plant is surrounded by any zombie (in all the 8 adjacent cells). If the plant is safe then update **count = count + 1**. Print the **count** in the end.

GeeksforGeeks

DAY 4

Find the non decreasing order array from given array

Given an array **A[]** of size **N / 2**, the task is to construct the array **B[]** of size N such that:

1. B[] is sorted in non-decreasing order.
2. A[i] = B[i] + B[n – i + 1].

Note: *Array **A[]** is given in such a way that the answer is always possible.*

Examples:

Input: *A[] = {3, 4}*
Output: *0 1 3 3*
Input: *A[] = {4, 1}*
Output: *0 0 1 4*

Approach: Let's present the following greedy approach. The numbers will be restored in pairs **(B[0], B[n – 1])**, **(B[1], B[n – 2])** and so on. Thus, we can have some limits on the values of the current pair (satisfying the criteria about sorted result).
Initially, **l = 0** and **r = 10^9**, they are updated with **l = a[i]** and **r = a[n – i + 1]**. Let **l** be minimal possible in the answer. Take **a[i] = max(l, b[i] – r)** and **r = b[i] – l**, that way **l** was chosen in such a way that both **l** and **r** are within the restrictions and **l** is also minimal possible.
If **l** was any greater than we would move both **l** limit up and **r** limit down leaving less freedom for later choices.

GeeksforGeeks

DAY 5

Count number of trailing zeros in (1^1)*(2^2)*(3^3)*(4^4)*..

Given an integer **n**, the task is to find the number of trailing zeros in the function

i.e. **f(n) = $1^1 * 2^2 * 3^3 * \ldots * n^n$.**

$$f(n) = \prod_{i=1}^{n} i^i$$

Examples:

Input: *n = 5*
Output: *5*
*f(5) = $1^1 * 2^2 * 3^3 * 4^4 * 5^5$ = 1 * 4 * 27 * 256 * 3125 = 86400000*
Input: *n = 12*
Output: *15*

Approach: We know that **5 * 2 = 10** i.e. 1 trailing zero is the result of the multiplication of a single 5 and a single 2. So, if we have **x** number of **5** and **y** number of **2** then the number of trailing zeros will be **min(x, y)**.
Now, for every number **i** in the series, we need to count the number of **2** and **5** in its factors say **x** and **y** but the number of **2s** and **5s** will be **x * i** and **y * i** respectively because in the series **i** is raised to the power itself i.e. i^i. Count the number of **2s** and **5s** in the complete series and print the minimum of them which is the required answer.

GeeksforGeeks

DAY 6

Partition negative and positive without comparison with 0

Given an array of n integers, both negative and positive, partition them into two different arrays without comparing any element with 0.

Examples:

```
Input : arr[] = [1, -2, 6, -7, 8]

Output : a[] = [1, 6, 8]
         b[] = [-2, -7]
```

Approach:

1. Initialize two empty vectors. Push the first element of the array in any of the two vectors, suppose the first vector. Let it be denoted by x
2. For every other element, arr[1] to arr[n-1], check if its sign and the sign of x is same or not. If the signs are the same, then push the element in the same vector. Else, push the element in the other vector
3. After the traversal of the two vectors has completed, print both the vectors

How to check if their signs are opposite or not?

Let the integers to be checked be denoted by x and y. The sign bit is 1 in negative numbers, and 0 in positive numbers. The XOR of x and y will have the sign bit as 1 if and only if they have opposite signs. In other words, XOR of x and y will be a negative number if and only if x and y have opposite signs.

GeeksforGeeks

DAY 7

Time taken by two persons to meet on a circular track

Given integers **L**, **S1** and **S2** where **L** is the length of a circular track in meters, **S1** and **S2** are the speeds of two persons in kilometers/hour moving in the same direction on the given track starting from the same starting point. The task is to find the following:

- The time after which they will meet for the first time.
- The time at which there are going to meet at the starting point.

Examples:

Input: *L = 30, S1 = 5, S2 = 2*
Output: *Met first time after 10 hrs*
Met at starting point after 30 hrs
Input: *L = 10, S1 = 1, S2 = 2*
Output: *Met first time after 10 hrs*
Met at starting point after 10 hrs

Approach:

- For calculating the time at which they will first meet.
  - First of all, calculate the Relative speed i.e. **S1 – S2**.
  - Then use the formula, **Time = Distance / Relative speed**.

- For calculating the time at which they will again meet at starting point.
  - First of all, calculate the time i.e. **T1** and **T2** which represent the time taken by both to cover 1 round of circular track by using the formula **Time = Length of track / Speed**.
  - Then calculate the LCM to know the time they will again meet at starting point.