DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DOMAIN WINTER WINNING CAMP

StudentName:Divya Sharma **UID:**22BCS14192
Branch:CSE **Section/Group:**FL_IOT-603/B

VERY EASY

1. N-th Tribonacci Number

The Tribonacci sequenceTn is defined as follows:
T0= 0,T1 = 1,T2 = 1, andTn+3 =Tn +Tn+1 +Tn+2 for n >= 0.
Givenn, return the value of Tn.

Example 1:
Input: n = 4
Output: 4
Explanation:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4

Example2:Input:n= 25
Output:1389537
Constraints:0<= n<= 37
The answer is guaranteed to fit with in a 32-bit integer, ie. answer<=2^31-1.

CODE:

```
Def tribonacci(n:int)->int: if
   n == 0:
       return0
   if n in (1, 2):
       return1
   dp = [0, 1, 1]
    fori in range(3,n + 1):
      dp.append(dp[i-1]+dp[i-2]+dp[i-3]) return
   dp[n]
print(tribonacci(4))
```

```
Output

4

=== Code Execution Successful ===
```

Easy

1. Climbing Stairs

You are climbing a staircase.It takes n steps to reach the top.Each time you can either climb1or 2 steps. In how many distinct ways can you climb to the top?

Example1:Input:n= 2
Output:2
Explanation:There are two ways to climb to the top.
step+ 1 step
2. 2 steps
Constraints:1<=n<=45

CODE:

```
Def climbStairs(n:int)->int: if
   n == 1:
      return1
   dp = [0] * (n + 1)
   dp[1], dp[2] = 1, 2
   fori in range(3,n + 1):
      dp[i]=dp[i-1]+dp[i-2] return
   dp[n]

print(climbStairs(2))#Output:2
```

```
Output

2

=== Code Execution Successful ===
```

***Medium*:**

2. Longest Palindromic Substring

Given a string s,return thelongest palindromic substring in s.

Example1:Input:s= "babad"

Output:"bab"
Explanation:"aba"isalsoavalidanswer.

Example2:Input:s= "cbbd"
Output:"bb"

Constraints:1<=s.length<=1000
sconsistofonlydigitsandEnglish letters.

CODE:

```
Def longestPalindrome(s:str)->str:
    defexpand_around_center(left,right):
        whileleft>=0andright<len(s)ands[left]==s[right]: left -= 1
            right += 1
        returnleft + 1,right - 1

    start,end =0, 0
    for i in range(len(s)):
        l1,r1=expand_around_center(i,i)
        l2,r2=expand_around_center(i,i+1) if
        r1 - l1 > end - start:
            start, end = l1,
        r1ifr2-l2>end-start:
            start,end=l2,r2
    returns[start:end+1]

print(longestPalindrome("babad"))#Output:"bab"or"aba"
```

```
Output
bab

=== Code Execution Successful ===
```

Hard

3.Maximal Rectangle

Given arowsxcols binary matrix filled with 0's and 1's,find the largest rectangle containing only 1's and return its area.

Example-

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

Input:matrix=[["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"], ["1","0","0","1","0"]]
Output:6
Explanation:Themaximal rectangle is shown in the above picture.

Constraints:
rows == matrix.length
cols==matrix[i].length 1
<= row, cols <= 200
matrix[i][j]is '0' or'1'.

CODE:

```
defmaximalRectangle(matrix):
    if not matrix:
        return0
    deflargest_histogram_area(heights):
        stack = []
        max_area = 0
        heights.append(0)
        for i, h in enumerate(heights):
            whilestackandheights[stack[-1]]>h: height
                = heights[stack.pop()]
                width = i if not stack else i - stack[-1] - 1
                max_area=max(max_area,height*width)
            stack.append(i)
        return max_area
    cols=len(matrix[0])
    heights = [0] * cols
    max_area = 0
    forrowin matrix:
        for j in range(cols):
            heights[j]=heights[j] +1 ifrow[j]== "1"else0
        max_area=max(max_area,largest_histogram_area(heights))
    return max_area
```

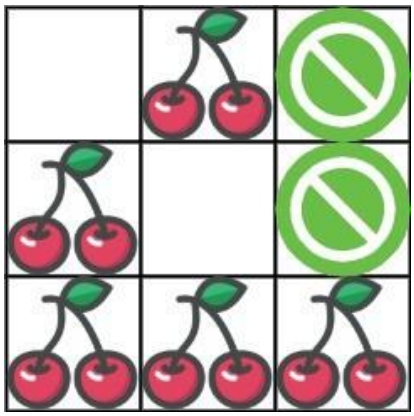
```
matrix=[["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],
["1","0","0","1","0"]]
print(maximalRectangle(matrix))
```

```
Output

6

=== Code Execution Successful ===
```

VeryHard

4.CherryPickup

Youaregivenannxngridrepresentingafieldofcherries,eachcellisoneofthreepossible integers.
0meansthecellisempty,soyoucanpass through,
1meansthecellcontainsacherrythatyoucanpickupandpassthrough,or
-1meansthecellcontainsathornthatblocksyourway.
Returnthemaximumnumberofcherriesyoucancollectbyfollowingtherulesbelow:
Starting at the position (0, 0) and reaching (n - 1, n - 1) by moving right or down through validpath cells (cells with value 0 or 1).
After reaching (n - 1, n - 1), returning to (0, 0) by moving left or up through valid path cells.
Whenpassingthroughapathcellcontainingacherry,youpickitup,andthecellbecomesan empty cell 0.
Ifthereis novalidpath between(0, 0)and(n -1,n -1), thennocherries canbecollected.



Input:grid = [[0,1,-1],[1,0,-1],[1,1,1]]
Output:5
Explanation:Theplayerstartedat(0,0)andwentdown,down,rightrighttoreach(2,2).

4 cherries were picked up during this single trip, and the matrix becomes [[0,1,-1],[0,0,-1], [0,0,0]].
Then,theplayerwentleft,up,up,lefttoreturnhome,pickinguponemorecherry. The total number of cherries picked up is 5, and this is the maximum possible.

Example2:Input:grid=[[1,1,-1],[1,-1,1],[-1,1,1]]
Output:0

Constraints:
n == grid.length
n==grid[i].length 1
<= n <= 50
grid[i][j]is -1, 0, or 1.
grid[0][0]!= -1
grid[n-1][n-1]!= -1

CODE:

```
defcherryPickup(grid):
    n = len(grid)
    dp=[[[float('-inf')]*nfor_inrange(n)]forinrange(n)]
    dp[0][0][0] = grid[0][0]

    for r1 in range(n):
        forc1inrange(n):
            for c2 in range(n):
                r2= r1 + c1 - c2
                if0<=r2<nandgrid[r1][c1]!=-1andgrid[r2][c2]!=-1: cherries =
                    grid[r1][c1]
                    ifc1 != c2:
                        cherries+=grid[r2][c2] prev
                    = max(
                        dp[r1-1][c1][c2] ifr1> 0elsefloat('-inf'),
                        dp[r1][c1-1][c2]if c1>0elsefloat('-inf'),
                        dp[r1-1][c1][c2-1]ifr1>0andc2>0elsefloat('-inf'), dp[r1][c1-1][c2 -
                        1] ifc1 >0and c2> 0elsefloat('-inf'),
                    )
                    dp[r1][c1][c2]=prev+cherriesifprev!=float('-inf')elsefloat('-inf') return

    max(0, dp[n - 1][n - 1][n - 1])

grid = [[0, 1, -1], [1, 0, -1], [1, 1, 1]]
print(cherryPickup(grid))
```

Output

```
0

=== Code Execution Successful ===
```