WEEK – 9

Aim : Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

Example 1:

Input: n = 3

Output: ["((()))","(()())","(())()","()(())","()()()"]

Example 2:

Input: n = 1

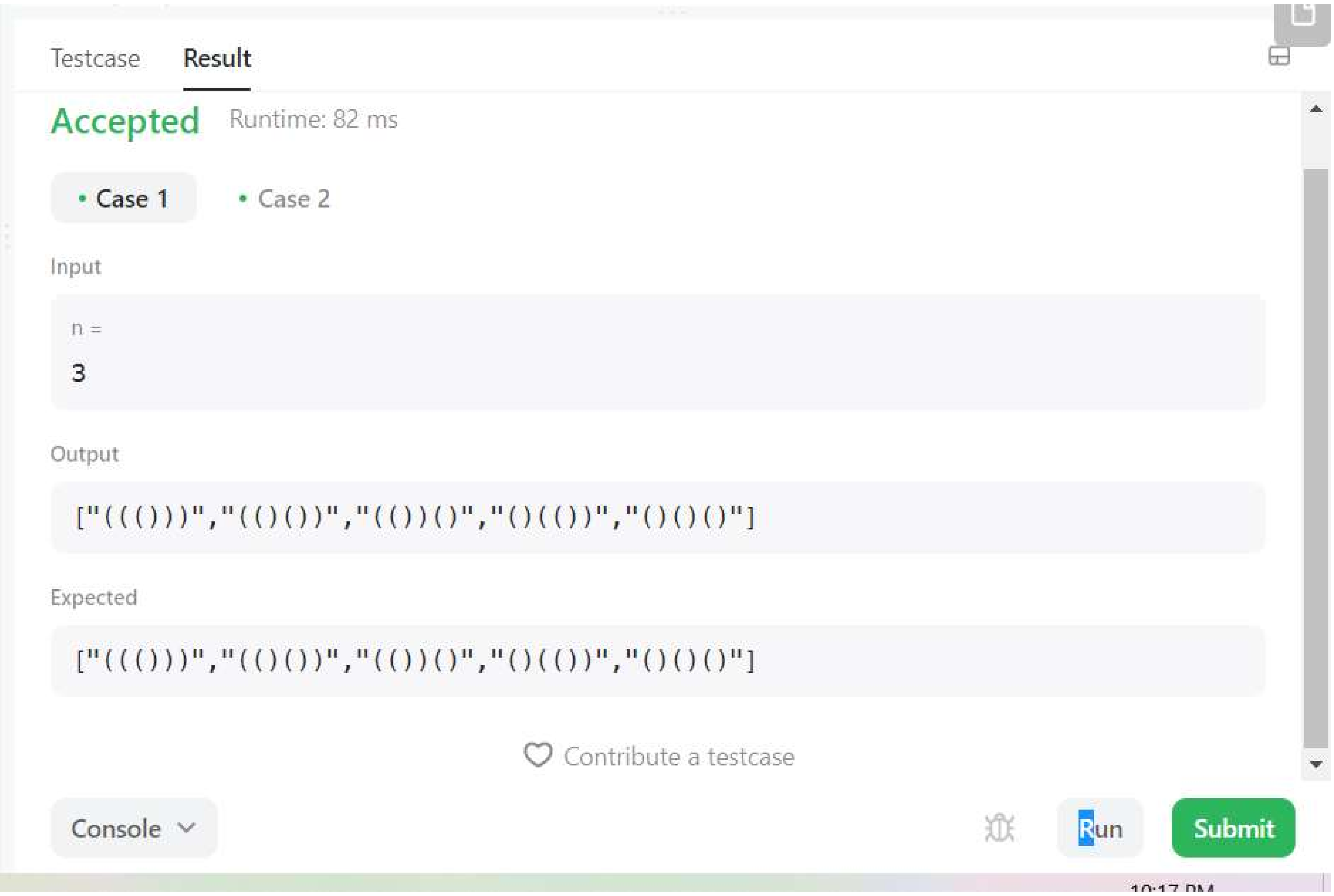
Output: ["()"]

Program :

```
class Solution:
    def generateParenthesis(self, n: int) -> List[str]:
        # Memo to store the already visited combinaison of the rest of parentheses
to open (named r for rest) and already opened parentheses (named o for
opened)
        memo = [[[] for _ in range(n + 1)] for _ in range(n + 1)]
        # Base value
        memo[0][0] = ["", ]
        memo[0][1] = [")",]
        memo[1][0] = ["()",]
        def f(r, o): # r  rest  o opened
            if memo[r][o]:
                return memo[r][o]
            if o == 0:  # if there is no resting parentheses, then close the ones already
openend
```

```
            memo[r][o] = ["("+e for e in f(r-1, o+1)]
        elif r == 0:  # if there is no opened parentheses, then open one
            memo[r][o] = [")"+e for e in f(r, o-1)]
        else:  # If there is opened and resting combinaison, then it is the
combinaison of two
            memo[r][o] = ["("+e for e in f(r-1, o+1)] + [")"+e for e in f(r, o-1)]
        return memo[r][o]
    return f(n, 0)
```

Output :



Accepted Runtime: 82 ms

Input

```
n =
3
```

Output

```
["((()))","(()())","(())()","()(())","()()()"]
```

Expected

```
["((()))","(()())","(())()","()(())","()()()"]
```

Result : Sucessfully Executed The program.

Aim : There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time.

Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.

The test cases are generated so that the answer will be less than or equal to 2 * 109.

Example :

Input: m = 3, n = 2

Output: 3

Explanation: From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:

1. Right -> Down -> Down

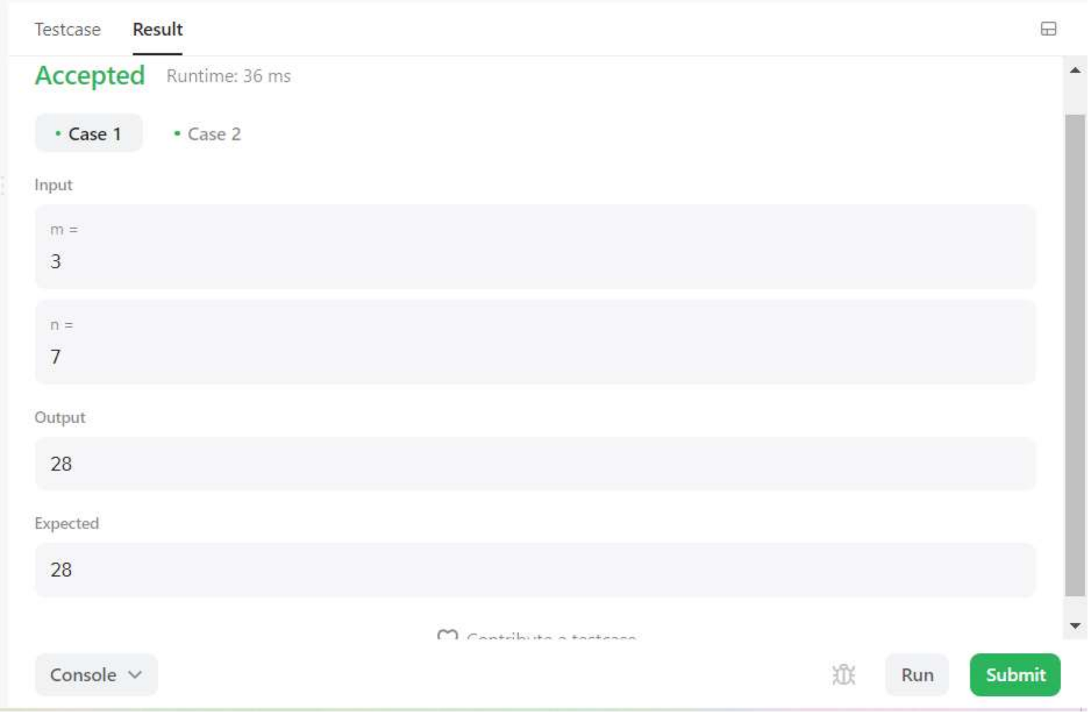
2. Down -> Down -> Right

3. Down -> Right -> Down

Program :

```
class Solution:
    def uniquePaths(self, m: int, n: int) -> int:
        memo=[[0 for _ in range(n+1)] for _ in range(m+1)]
        def db(x,y):
            if x==m and y==n:
                return 1
            elif x==m:
                if memo[x][y+1]==0:
                    memo[x][y+1]=db(x,y+1)
        return memo[x][y+1]
    elif y==n:
        if memo[x+1][y]==0:
            memo[x+1][y]=db(x+1,y)
        return memo[x+1][y]
    else:
        if memo[x][y+1]==0:
            memo[x][y+1]=db(x,y+1)
        if memo[x+1][y]==0:
            memo[x+1][y]=db(x+1,y)
        return memo[x][y+1]+memo[x+1][y]
return db(1,1)
```

Output :



Accepted Runtime: 36 ms

Input

m =
3

n =
7

Output

28

Expected

28

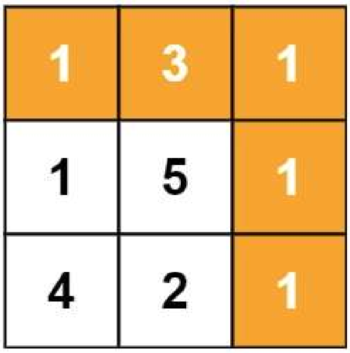
Result : Sucessfully Executed The program.

WEEK – 10

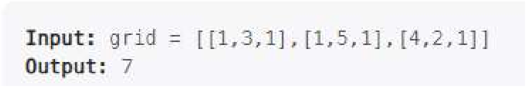
Aim : Implement minimum path sum.

Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

Note: You can only move either down or right at any point in time.



1	3	1
1	5	1
4	2	1

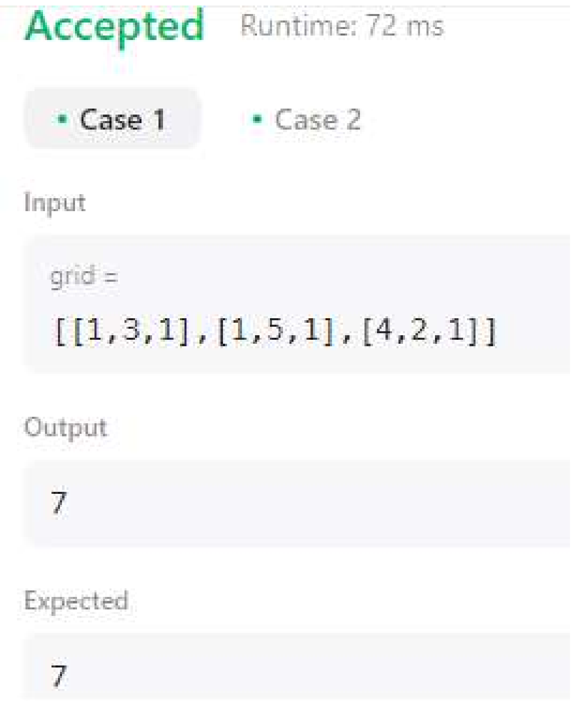


```
Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
```

Program :

```
class Solution:
    def minPathSum(self, grid: List[List[int]]) -> int:
        m, n = len(grid), len(grid[0])
        for i in range(1, m):
            grid[i][0] += grid[i-1][0]
        for i in range(1, n):
            grid[0][i] += grid[0][i-1]
        for i in range(1, m):
            for j in range(1, n):
                grid[i][j] += min(grid[i-1][j], grid[i][j-1])


        return grid[-1][-1]
```

Output :



Accepted Runtime: 72 ms

Input

grid =

```
[[1,3,1],[1,5,1],[4,2,1]]
```

Output

```
7
```

Expected

```
7
```

Result : Sucessfully Executed The program.

Aim : Implement Counting Bits.

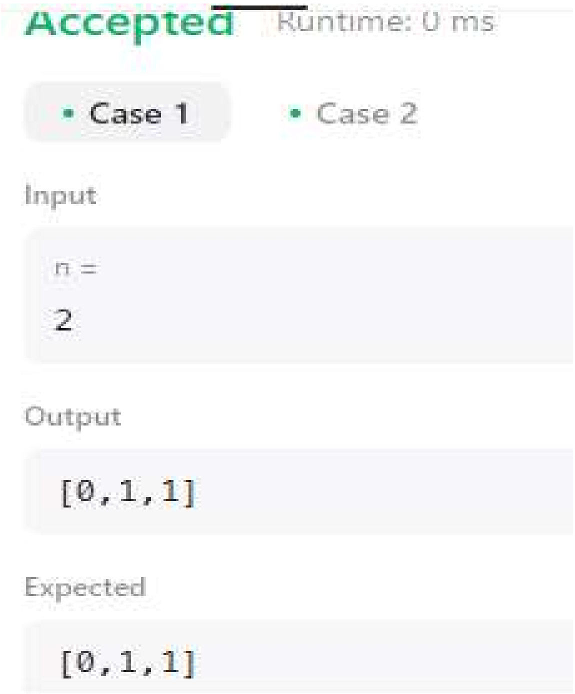
Given an integer n, return an array ans of length n + 1 such that for each i (0 <= i <= n), ans[i] is the number of 1's in the binary representation of i.

Program :

```
class Solution
{
  public int[] countBits(int n)
  {
    int[] a = new int[n+1];
    a[0]=0;
    if(n>0){a[1]=1;}
    if(n>1){
      int sum=2;
      for (int i =2;1<2;i*=2 ){
        for (int j=0;j<i;j++){
          a[i+j]=a[j]+1;
            sum++;
          if(sum==n+1){
            break;
          }
        }


        if(sum==n+1){
          break;
        }
```

```
            }
        }
    return a;
    }
}
```

Output :



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

n =

2

Output

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

Expected

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

Result : Sucessfully Executed The program.

WEEK – 11

Aim : Implement Regular Expression Matching.

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

'.' Matches any single character.

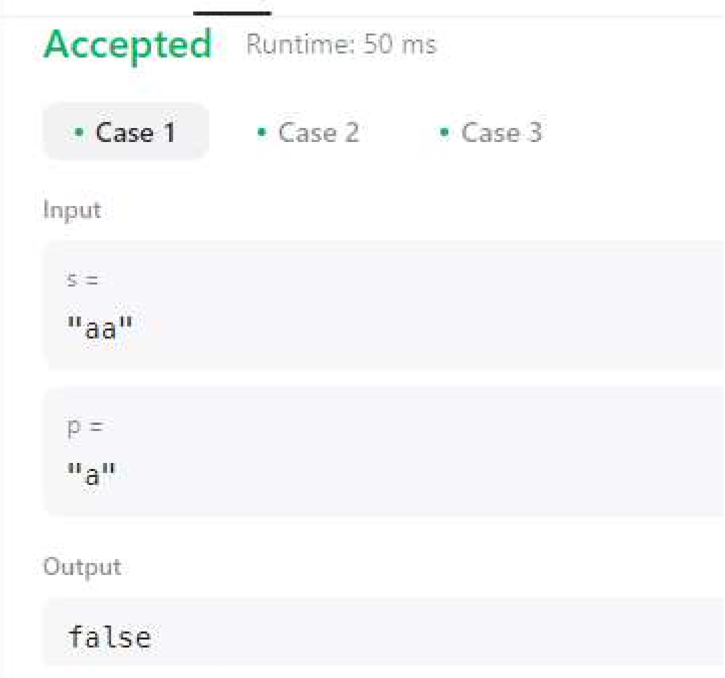
'*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

Program ;

```
class Solution:
    @lru_cache()
    def isMatch(self, s: str, p: str) -> bool:

        if not p: return not s

        x = bool(s) and p[0] in {'.',s[0]}

        if len(p)>=2 and p[1] == '*':

            return self.isMatch(s,p[2:]) or (x and self.isMatch(s[1:],p))

        else:

            return x and self.isMatch(s[1:],p[1:])
```

Output :



Accepted Runtime: 50 ms

Input

```
s =
"aa"
```

```
p =
"a"
```

Output

```
false
```

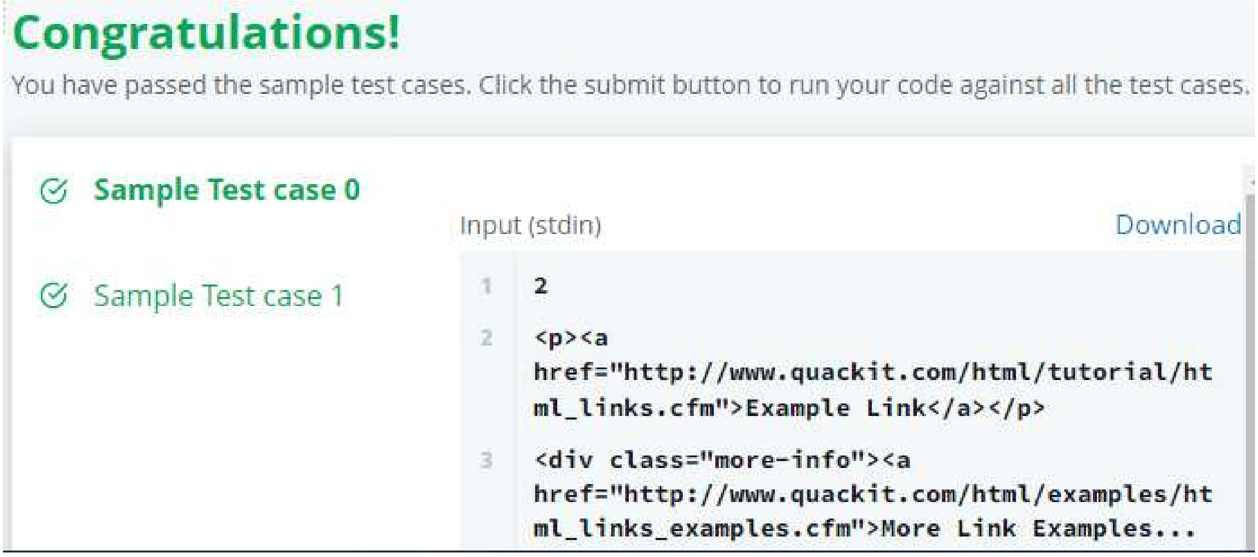
Result : Sucessfully Executed the Program.

Aim : Implement Program to Detect HTML Links in given input.

Program :

```
import re
for i in range(int(input().strip())):
   data = input().strip()
   matches = re.findall(r'[^<]*<a href="([^"]+)".*?>(?:[^<]<\w+>)*([^<]*?)(?:<\/\w+>)*<\/a>', data)
   if matches:
      for m in matches:
         print("{0},{1}".format(m[0].strip(), m[1].strip()))
```

Output:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

Input (stdin)

Download

```
2
<p><a href="http://www.quackit.com/html/tutorial/html_links.cfm">Example Link</a></p>
<div class="more-info"><a href="http://www.quackit.com/html/examples/html_links_examples.cfm">More Link Examples...
```

Result : Sucessfully Executed The Program.

WEEK – 12

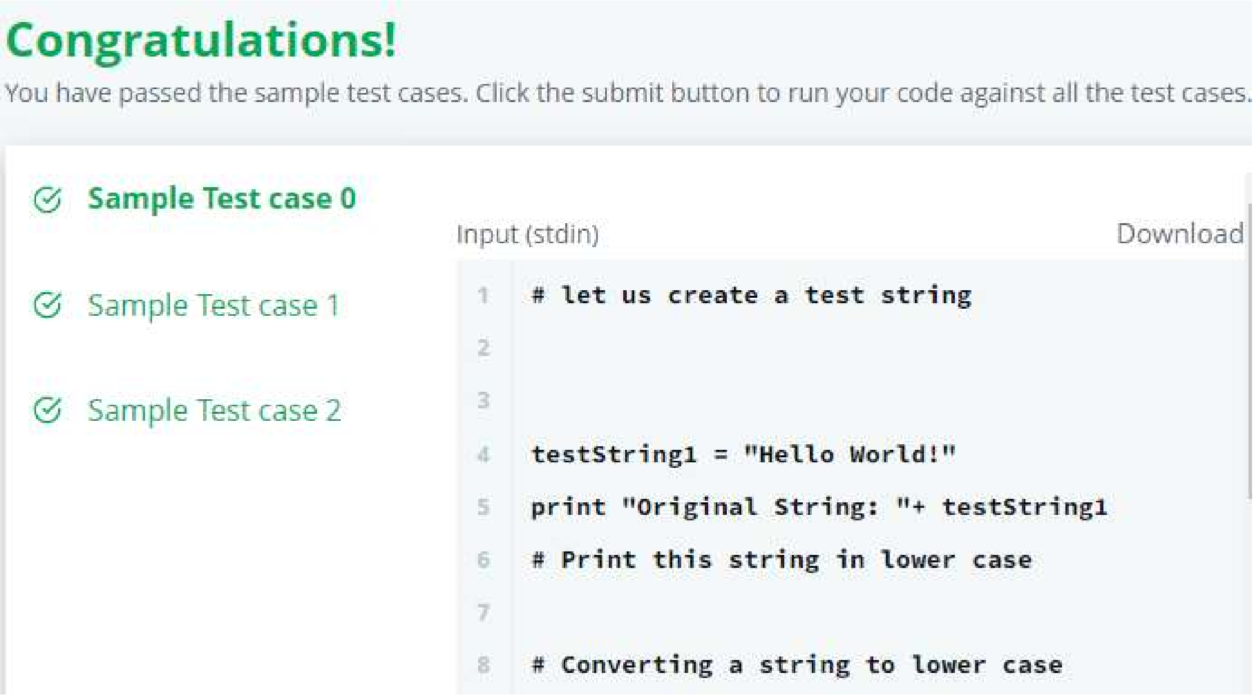
Aim : Building a Smart IDE: Programming Language Detection

We are trying to hack together a smart programming IDE. Help us build a feature which auto-detects the programming language, given the source code. There are only three languages which we are interested in "auto-detecting": Java, C and Python.

Program :

```
import re
from sys import stdin

java = r'public class|java\.io'
c = r'#include'

code = ''.join(stdin.read())

if re.search(java, code):
   print('Java')

elif re.search(c, code):
   print('C')

else:
   print('Python')
```

Output:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

✓ Sample Test case 2

```
# let us create a test string



testString1 = "Hello World!"
print "Original String: "+ testString1
# Print this string in lower case

# Converting a string to lower case
```

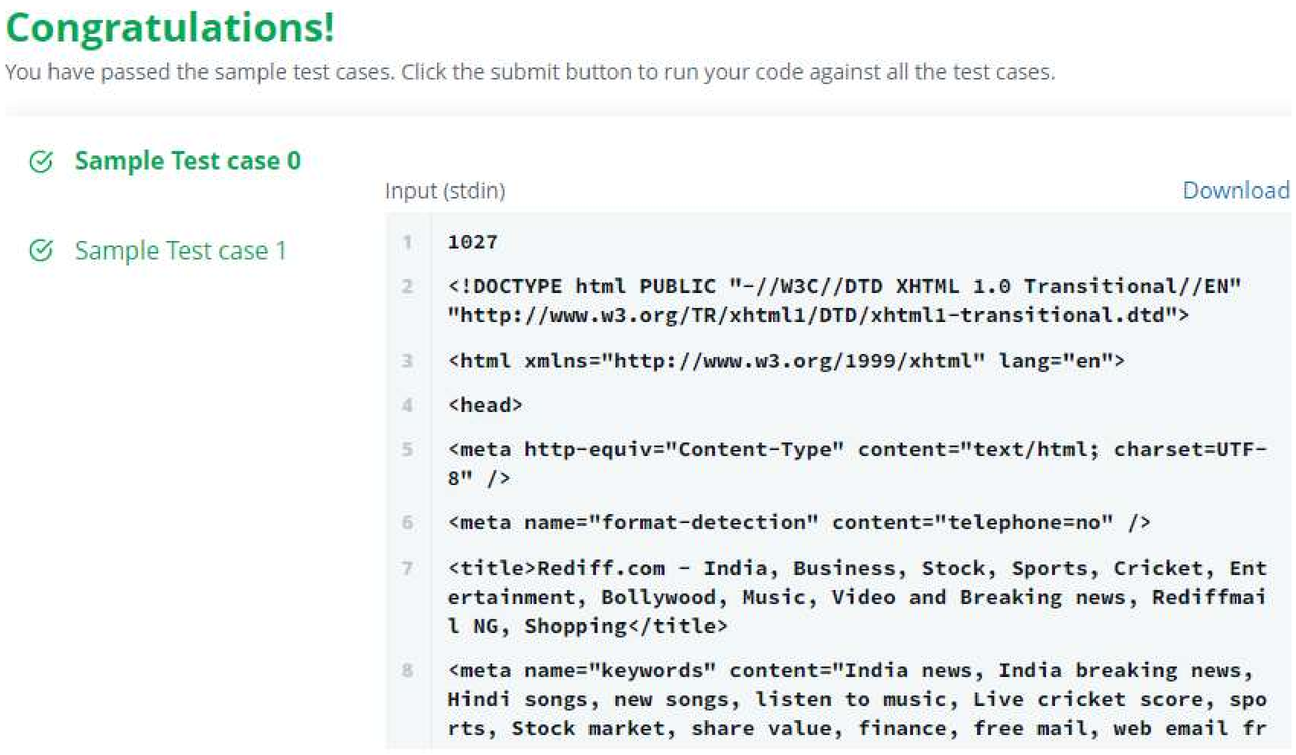
Result : Sucessfully Executed The program.

Aim : : Implement Program to Detect Domain names.

Program :

```
import re

txt='\n'.join([input() for _ in range(int(input()))])

print(*sorted(set(re.findall(r'https?://(?:ww(?:w|2)\.)?([\w\.\-]*\.[a-zA-Z]+)',txt,re.DOTALL))),sep=';')
```

Output :



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

Input (stdin)

```
1027
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="format-detection" content="telephone=no" />
<title>Rediff.com - India, Business, Stock, Sports, Cricket, Entertainment, Bollywood, Music, Video and Breaking news, Rediffmail NG, Shopping</title>
<meta name="keywords" content="India news, India breaking news, Hindi songs, new songs, listen to music, Live cricket score, sports, Stock market, share value, finance, free mail, web email fr
```

Result : Sucessfully Executed the program.