**Week – 9**

**Aim:** Best Time to Buy and Sell Stock.

**Explanation**:

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

**Code:**

class Solution {

public:

    int maxProfit(vector<int>& p) {

        int minimum=p[0];

        int ans=0;

        for(int i=1;i<p.size();i++){

          ans=max(ans,p[i]-minimum);

          minimum=min(minimum,p[i]);

        }

        return ans;

    }

};

**Output:**



**Result:** The given program was executed successfully.

**Aim:** Implement Palindrome Partitioning.

**Explanation**:

Given a string s, partition s such that every Substring of the partition is a palindrome Return all possible palindrome partitioning of s.

**Code:**

class Solution {

public:

    bool checkPalindrome(string str, int startIndex, int lastIndex){

        while(startIndex <= lastIndex){

            if(str[startIndex] != str[lastIndex])

                return false;

            startIndex++;

            lastIndex--;

        }

        return true;

    }

    void palindromePartition(int index, vector<string>& ds, vector<vector<string>>& output, string str){

        if(index == str.length()){

            output.push\_back(ds);

            return;

        }

        for(int i = index; i < str.length(); i++){

            if(checkPalindrome(str, index, i)){

                ds.push\_back(str.substr(index, i - index + 1));

                palindromePartition(i+1, ds, output, str);

                ds.pop\_back();

            }

        }

    }

    vector<vector<string>> partition(string s) {

        vector<vector<string>> output;

        vector<string> ds;

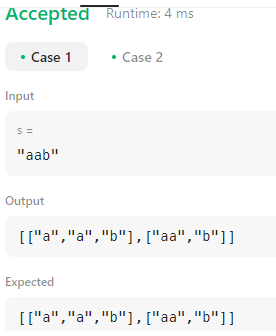
        palindromePartition(0, ds, output, s);

        return output;

    }

};

**Output:**



**Result:** The given program was executed successfully.

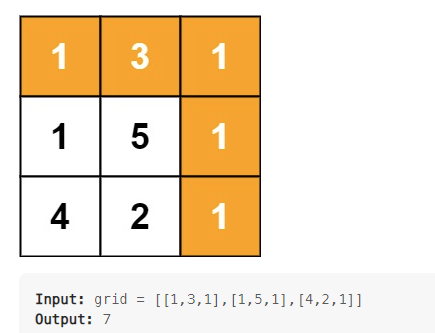
**Week – 10**

**Aim:** Implement minimum path sum.

**Explanation**:

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.



**Code:**

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]

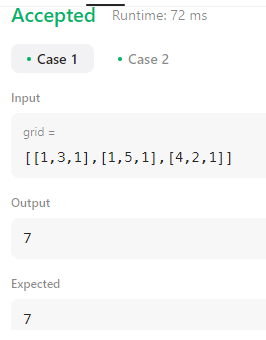
        for i in range(1, m):

            for j in range(1, n):

                grid[i][j] += min(grid[i-1][j], grid[i][j-1])

        return grid[-1][-1]

**Output:**



**Result:** The given program was executed successfully.

**Aim:** Implement Counting Bits.

**Explanation**:

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.

**Code:**

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;

                }

            }

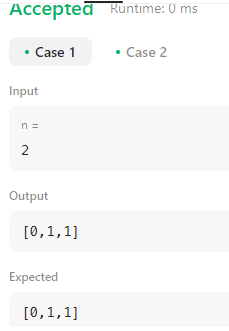
        }

    return a;

    }

}

**Output:**



**Result:** The given program was executed successfully.

**Week – 11**

**Aim:** Implement Regular Expression Matching.

**Explanation**:

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).

**Code:**

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]}

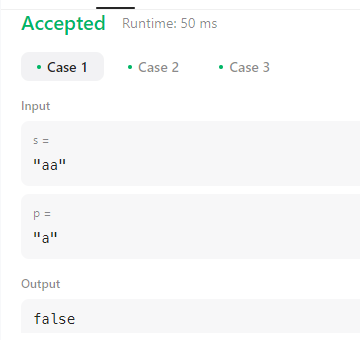
        if len(p)>=2 and p[1] == '\*':

            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:**



**Result:** The given program was executed successfully.

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

**Code:**

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:**



**Result:** The given program was executed successfully.

**Week – 12**

**Aim:** Building a Smart IDE: Programming Language Detection

**Explanation**:

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.

**Code:**

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')

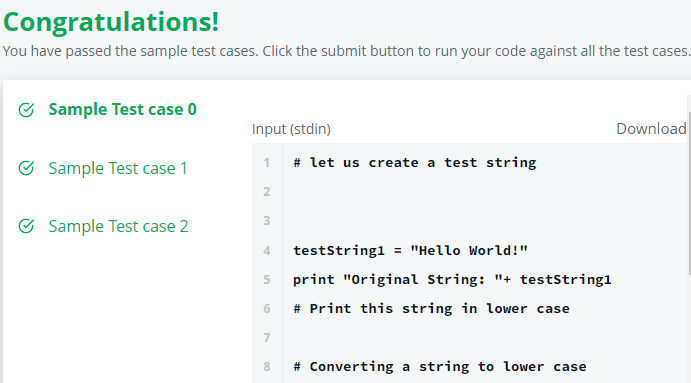
elif re.search(c, code):

    print('C')

else:

    print('Python')

**Output:**



**Result:** The given program was executed successfully.

**Aim:** Implement Program to Detect Domain names.

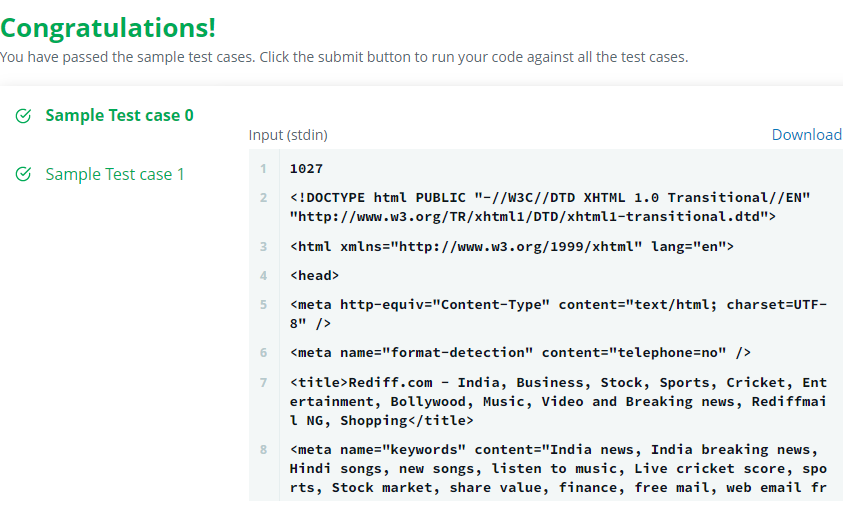
**Code:**

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:**



**Result:** The given program was executed successfully.