

ELP 780  
SOFTWARE LAB



INDIAN INSTITUTE OF TECHNOLOGY  
NEW DELHI

---

Assignment 8

---

September 27, 2018

*Name:* **Priya Kumari**

*Entry Number:*  
**2017EET2305**

---

# Contents

<b>1</b>	<b>Problem Statement 1</b>	<b>1</b>
1.1	Assumptions . . . . .	1
1.2	Program Structure . . . . .	1
1.3	Flowchart . . . . .	2
1.4	Constraints . . . . .	3
1.5	Input . . . . .	3
1.6	Output . . . . .	3
1.7	Input Screenshot . . . . .	3
1.8	Output Screenshots . . . . .	3
<b>2</b>	<b>Problem Statement 2</b>	<b>4</b>
2.1	Assumptions . . . . .	4
2.2	Program Structure . . . . .	4
2.3	Flowchart . . . . .	5
2.4	Output Format . . . . .	6
2.5	Input Format . . . . .	6
2.6	Output Format . . . . .	6
2.7	Input Screenshot . . . . .	6
2.8	Output Screenshot . . . . .	6
<b>3</b>	<b>Code</b>	<b>7</b>
<b>4</b>	<b>Bibliography</b>	<b>10</b>

# 1 Problem Statement 1

Question 1: IIT Delhi, has just got the strongest computer. The professors in charge wants to check the computational capacity of the computer. So, they decided to create the problem which is to be given as an assignment to students. Can you help the professor to check the computation capability of the computer?

A valid cross is defined here as the two regions (horizontal and vertical) of equal lengths crossing over each other. These lengths must be odd, and the middle cell of its horizontal region must cross the middle cell of its vertical region.

Find the two largest valid crosses that can be drawn on smart cells in the grid, and return two integers denoting the dimension of the each of the two largest valid crosses. In the above diagrams, our largest crosses have dimension of 1, 5 and 9 respectively .

## 1.1 Assumptions

The two crosses cannot overlap, and the dimensions of each of the valid crosses should be maximal.

## 1.2 Program Structure

1. take the values of k1 l2 k3
2. take the string
3. divide string into group1 group2 group3
4. rotate the 3 strings
5. replace to make result

### 1.3 Flowchart

## 1.4 Constraints

$2 \leq n \leq 105$

$2 \leq m \leq 105$

## 1.5 Input

### Input format

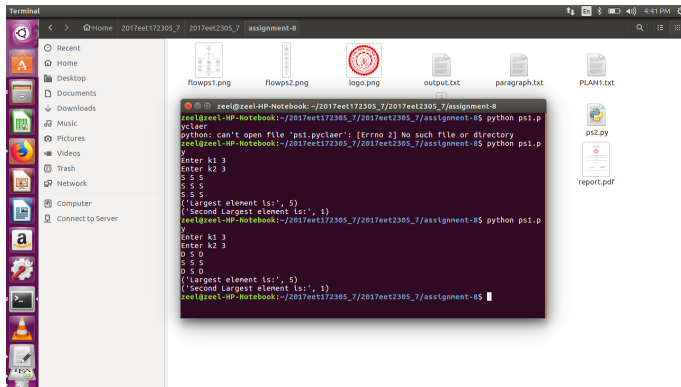
The first line contains two space-separated integers,  $n$  and  $m$ . Each of the next  $n$  lines contains a string of  $m$  characters where each character is either  $S$  (Smart) or  $D$  (Dull). These strings represent the rows of the grid. If the  $j$ th character in the  $i$ th line is  $S$ , then  $(i,j)$  is a cell smart. Otherwise it's a dull cell.

## 1.6 Output

### Input format

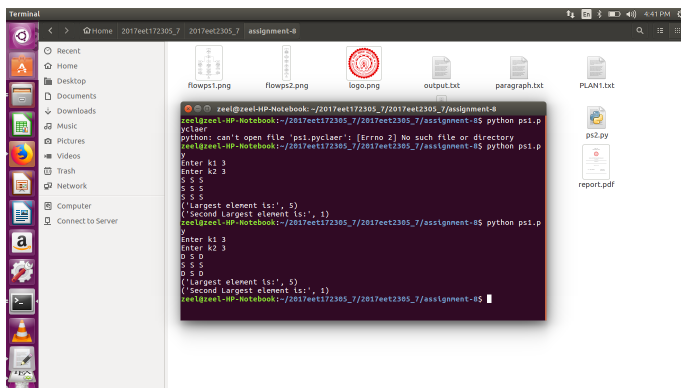
Find two valid crosses that can be drawn on smart cell of the grid, and return the dimension of both the crosses in the reverse sorted order(i.e. First Dimension should be the larger one and other should be smaller one).

## 1.7 Input Screenshot



```
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$ python ps1.py
python: can't open file 'ps1.py': [Errno 2] No such file or directory
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$ python ps1.py
Enter k1 5
Enter k2 5
S S S
S S S
S S S
('Largest element is:', 5)
('Second Largest element is:', 1)
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$ python ps1.py
Enter k1 3
Enter k2 3
S S D
S S D
S S D
('Largest element is:', 5)
('Second Largest element is:', 1)
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$
```

## 1.8 Output Screenshots



```
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$ python ps1.py
python: can't open file 'ps1.py': [Errno 2] No such file or directory
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$ python ps1.py
Enter k1 3
Enter k2 3
S S D
S S D
S S D
('Largest element is:', 5)
('Second Largest element is:', 1)
zeel@zeel-HP-Notebook:~/2017feet172305_7/2017feet2305_7/assignment-8$
```

## 2 Problem Statement 2

Question 2: After, getting mix results of valid crosses, professors decided to test the computation abilities on one more problem. This time professors wanted to test the decryption capabilities of the computer.

Encryption of a message requires three keys,  $k_1$ ,  $k_2$ , and  $k_3$ . The 26 letters of English and underscore are divided in three groups, [a-i] form one group, [j-r] a second group, and everything else ([s-z] and underscore) the third group. Within each group the letters are rotated left by  $k_i$  positions in the message. Each group is rotated independently of the other two. Decrypting the message means doing a right rotation by  $k_i$  positions within each group.

### 2.1 Assumptions

Rotating letters in one group will not change any letters in any of the other groups.

### 2.2 Program Structure

1. enter rows and cols from user
2. enter the S D matrix
3. define a structure with left right up down values
4. iterate through each cell and update the left right top down
5. store the values in a list
6. find max and second max

## 2.3 Flowchart

## 2.4 Output Format

1  $j$  = Length of the string  $j=150$

1 <sub>$i$</sub>  =  $k_i$   $j=150$  ( $i=1,2,3$ )

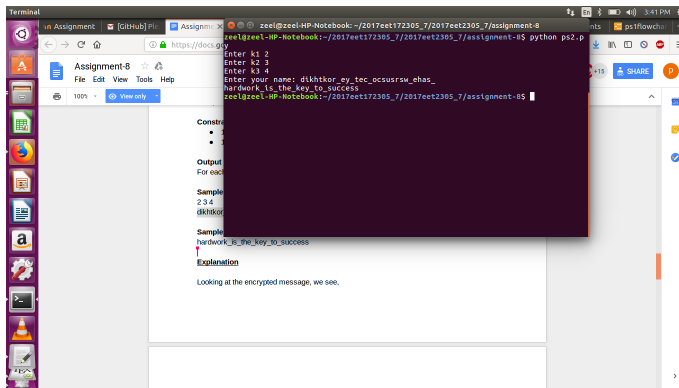
## 2.5 Input Format

All input strings comprises of only lowercase English alphabets and underscores

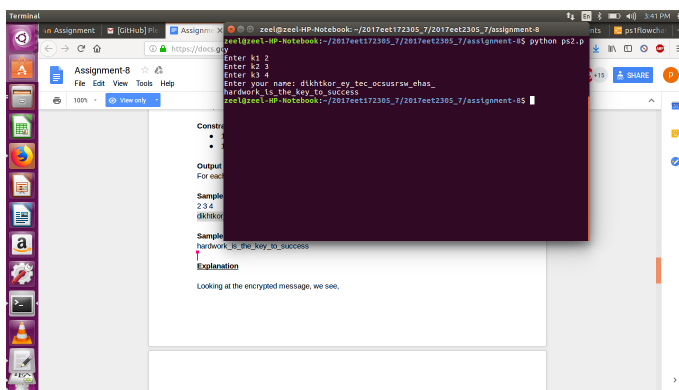
## 2.6 Output Format

For each encrypted message, the output is a single line containing the decrypted string.

## 2.7 Input Screenshot



## 2.8 Output Screenshot





### 3 Code

code1

```
##### this is the first .py file #####

##### write your code here #####

class cell:
    def __init__(self, char):
        self.char = char
        self.top = 0
        self.down = 0
        self.left=0
        self.right=0
        self.val=0;
        self.visit=0;
# find the largest and second largest elemet in the matrix
def Range(list1):
    if( len(list1) >0):
        largest = list1[0]
        largest2 = list1[0]
        for item in list1:
            if item > largest:
                largest = item
            elif largest2!=largest and largest2 < item:
                largest2 = item
        print("Largest_element_is:", largest*4+1)
        print("Second_Largest_element_is:", largest2*4+1)

m = int(input("Enter_k1_"))# enter k1
n = int(input("Enter_k2_"))# enter k2
matrix = []; columns = []
for i in range(0,m):
    matrix += [0]
# initialize the number of columns
for j in range (0,n):
    columns += [0]
# initialize the matrix
for i in range (0,m):
    matrix[i] = columns
for i in range (0,m):
    matrix[i]=raw_input().split("_") # input matrix row from user

array=[]
# definnd structure matrix
mat = [[0 for x in range(m)] for x in range(n)]
# initialize the new matrix
```

```

for i in range(m):
    for j in range(n):
        mat[i][j]=cell(matrix[i][j])
# assign the values of top left up down from the original matrix
for i in range(m):
    for j in range(n):
        if (mat[i][j].char == 'S'):
            if (i-1>=0 and mat[i-1][j].char == 'S'):
                mat[i][j].top=mat[i-1][j].top+1
            if (i+1<n and mat[i+1][j].char == 'S'):
                mat[i][j].down=mat[i+1][j].down+1
            if (j-1>=0 and mat[i][j-1].char == 'S'):
                mat[i][j].left=mat[i][j-1].left+1
            if (j+1<n and mat[i][j+1].char == 'S'):
                mat[i][j].right=mat[i][j+1].right+1
            mat[i][j].val=min(mat[i][j].left ,mat[i][j].right ,mat[i][
            array.append(mat[i][j].val)
# function call for the largest and second largest grid
Range(array)

code2

k1 = int(input("Enter_k1_"))# enter k1
k2 = int(input("Enter_k2_"))# enter k2
k3 = int(input("Enter_k3_"))# enter k3
name = raw_input("Enter_your_name:_")
group1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'] # group 1
group2 = ['j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r']# group 3
group3 = ['s', 't', 'u', 'v', 'w', 'x', 'y', 'z', '-']# group 3
s1=""
s2=""
s3=""
s = ""
# divide strings based on groups
for c in name:
    if c in group1:
        s1+=c
    elif c in group2:
        s2+=c
    elif c in group3:
        s3+=c

org1=s1
org2=s2
org3=s3
#rotate strings
s1=s1[-k1:]+s1[:-k1]
#s1=ss1+s1[:-k1]

```

```

#rotate strings
s2=s2[-k2:]+s2[: -k2]
#s2=ss2+s2[: -k2]

#rotate strings
s3=s3[-k3:]+s3[: -k3]
#s3=ss3+s3[: -k3]

#final result
result=""
i=0
j=0
k=0
#append strings in final result
for c in name:
    if c in s1:
        result+=s1[i]
        i=i+1
    elif c in s2:
        result+=s2[j]
        j=j+1
    elif c in s3:
        result+=s3[k]
        k=k+1
print(result)

```

## 4 Bibliography

1. <https://www.stackoverflow.com>
2. <https://code2flow.com>
3. <https://www.geeksforgeeks.org/python-maximum-minimum-elements-position-list/>
4. <https://stackoverflow.com/questions/33181350/quickest-way-to-find-the-nth-largest-value-in-a-numpy-matrix>

