Bài 1:

Minesweeper là một trò chơi cực kỳ phổ biến đã gắn liền với bao thế hệ người dùng máy tính. Ở trò chơi này, khi người dùng nhấn vào một ô nào đó trên bản đồ, một vùng liên tiếp sẽ được mở ra trong vùng đó mỗi ô sẽ mang một giá trị số tự nhiên cho biết trong các ô liền kề xung quanh ô đó (mỗi ô có tối đa 8 ô xung quanh) sẽ có bao nhiêu mìn.  
  
  
  
Vấn đề là các máy tính ngày xưa rất yếu nên nếu chờ người dùng nhấn rồi mới tính xem ở quanh mỗi ô có bao nhiêu mìn thì game bị lag. Do vậy giá trị mìn quanh mỗi ô được tính toán từ đầu game. Bạn hãy viết chương trình thực hiên tính toán đó.

INPUT  
Dòng đầu tiên chứa hai số tự nhiên m và n là kích thước sàn nhà của Bình (đơn vị là ô gạch)  
  
m dòng tiếp theo mỗi dòng chứa n số nguyên mang giá trị:  
  
Số 0 đại diện cho ô không có mìn  
Số 1 đại diện cho ô có mìn  
OUTPUT  
Xuất lại ma trận trên sau khi tính toán số mìn nằm xung quanh mỗi ô.

m,n = map(int,input().split())

ab = [[0 for j in range(n+2)] for i in range(m+2)]

for i in range(1,m+1):

l = list(map(int,input().split()))

for j in range(0,n):

ab[i][j+1] = l[j]

abr = [[0 for j in range(n)] for i in range(m)]

for i in range(1,m+1):

for j in range(1,n+1):

dem = 0

if ab[i][j] == 1:

if ab[i+1][j] == 1:

dem += 1

if ab[i - 1][j] == 1:

dem += 1

if ab[i][j+1] == 1:

dem += 1

if ab[i][j-1] == 1:

dem +=1

if ab[i+1][j+1] == 1:

dem += 1

if ab[i - 1][j - 1] == 1:

dem += 1

if ab[i-1][j+1] == 1:

dem+=1

if ab[i+1][j-1] == 1:

dem +=1

print(dem, end=' ')

elif ab[i][j] == 0:

if ab[i+1][j] == 1:

dem += 1

if ab[i - 1][j] == 1:

dem += 1

if ab[i][j+1] == 1:

dem += 1

if ab[i][j-1] == 1:

dem +=1

if ab[i+1][j+1] == 1:

dem += 1

if ab[i - 1][j - 1] == 1:

dem += 1

if ab[i-1][j+1] == 1:

dem+=1

if ab[i+1][j-1] == 1:

dem +=1

print(dem, end= ' ')

print()

Bài 2:

Viết hàm đảo mảng 2 chiều thành mảng đối xứng của nó theo trục ngang  
  
INPUT  
Dòng đầu tiên chứa hai số r và c là số dòng và số cột của mảng, mỗi số không quá 1000.  
  
r dòng tiếp theo, mỗi dòng chứa c số nguyên. Đây là các phần tử trong mảng 2 chiều  
  
OUTPUT  
Mảng sau khi đảo

m,n = map(int,input().split())

ab = [[0 for j in range(n)] for i in range(m)]

for i in range(0,m):

l = list(map(int,input().split()))

for j in range(0,n):

ab[i][j] = l[j]for i in range(m - 1,-1,-1):

for j in range(0,n):

print(ab[i][j], end= ' ')

print()

Bài 3:

Khoa học tiên tiến đã nảy sinh ra yêu cầu dùng máy tính để tự động phân tích ảnh chụp của các siêu viễn vọng kính nhằm phát hiện ra những vì tinh tú mới. Lý do cho nhu cầu này đơn giản là vì trên trời triệu triệu vì sao, đếm hoài đếm mải con người bó tay rồi, không đếm nổi.  
  
Các bạn hãy viết một chương trình nhận vào ảnh chụp của viễn vọng kính, ảnh này đã được xử lý đưa về dạng đen trắng. Điểm ảnh màu đen biểu diễn bằng ký tự # và điểm ảnh màu trắng biểu diễn bằng ký tự -  
  
Vùng màu đen trên ảnh là vũ trụ bao la rộng lớn còn vùng màu trắng chính là các ngôi sao. Một ngôi sao tùy kích thước to nhỏ có thể chiếm một điểm ảnh hoặc là nhiều điểm ảnh liền kê nhau theo đường ngang hoặc đường dọc (không tính đường chéo).  
  
Input  
Mỗi test case bắt đầu bằng một hàng chứa hai con số 1≤m,n≤100. Sau đó là m hàng, mỗi hàng chứa n điểm ảnh  
  
Output  
Ứng với mỗi test case, xuất ra số ngôi sao có trên ảnh theo ví dụ sau:

import sys

a = [l for l in sys.stdin.read().strip().split('\n')]

d = 0

case\_line = 0

while True:

m, n = map(int, a[case\_line].split())

temp = a[case\_line + 1:case\_line + m + 1]

temp = [[i for i in line] for line in temp]

def bucket\_fill(a,i,j):

if i < 0 or j < 0 or i >= len(a) or j >= len(a[0]):

return

if a[i][j] != '-':

return

a[i][j] = '!'

bucket\_fill(a,i - 1,j)

bucket\_fill(a,i + 1,j)

bucket\_fill(a,i,j - 1)

bucket\_fill(a,i,j + 1)

count = 0

for so\_hang, hang in enumerate(temp):

for so\_cot, cot in enumerate(hang):

if cot == '-':

count += 1

bucket\_fill(temp,so\_hang,so\_cot)

d += 1

print(f"Case {d}: {count}")

case\_line += m + 1

if case\_line >= len(a):

break

Bài 4:

Viết chương trình xuất ra màn hình mảng 2D kích thước n\*n với các số từ 1 đến n\*n được bố trí theo thứ tự xoắn ốc thuận chiều kim đồng hồ, bắt đầu từ ô ở góc trái trên.  
  
INPUT  
Số tự nhiên n  
  
OUTPUT  
Mỗi hàng của mảng trên một dòng, các số trong hàng cách nhau bởi 1 ký tự khoảng trắng

m = int(input())

ab = [[0 for j in range(m)] for i in range(m)]

a = []

for i in range(1,m\*m+1):

a.append(i)

i = 0

j = 0

l = 0

while i < m\*m :

for j in range(l,m - l):

ab[l][j] = a[i]

i+=1

for k in range(l + 1, m - l):

ab[k][m - l - 1] = a[i]

i += 1

for j in range(m - l - 2, l - 1,-1):

ab[m -l - 1][j] = a[i]

i += 1

for k in range(m - l - 2,l,-1):

ab[k][l] = a[i]

i+=1

l += 1

for i in range(0,m):

for j in range(0,m):

print(ab[i][j], end= ' ')

print()

Bài 5:

Some senior students under Dr. Thành supervision are working on self-driving autonomous robots. The movement system of the robots those students are working is very limited, the robot only understands one of the following movement command:move backward or forward for a fixed, small distance. They can turn either left or right but at exactly 90 degree. To move such robots to the desired destination, the navigation system is very important. The robot would use its camera to map its surrounding into a grid with the every point with integer coordinate in the grid represent a point the robot could reach after certain number of movement commands. The point that is obstructed by terrain will be marked as impassable. The navigation system will the plot a course for the robot as an array of point it has to go through to reach desired destination.  
  
  
  
Dr. Thành is reviewing the navigation system his senior students has just finished and right away, he can tell that something is off. The system seem to make more right turn than it should have. But as a renown computer scientists, Dr. Thành cannot draw a conclusion just based on intuition, he needs concrete data, even for just a small task like reviewing student’s work. So he writes a small program to count the number of right turn the navigation is asking the robot to make. The code of his program is given below, but he’s too busy and the code is not finished. It’s your turn to finish his code.  
  
INPUT  
The first line is the number of points n (2 <= n <= 10000) the navigation system was telling the robots to go through  
  
The next n lines are the coordinate (x, y) of n points respectively (x and y are space-separated)  
  
OUTPUT  
The nubmer of rights turns the robot would have to make.

n = int(input())

x = []

y = []

for i in range(0,n):

k = list(map(int,input().split()))

x.append(k[0])

y.append(k[1])

dem = 0

for i in range(2,n):

x1 = x[i-1] - x[i-2]

y1 = y[i-1] - y[i-2]

x2 = x[i] - x[i-1]

y2 = y[i] - y[i-1]

if y1 == 0:

if x1 > 0:

if x2 == 0 and y2 < 0:

dem += 1

elif x1 < 0:

if x2 == 0 and y2 > 0:

dem +=1

elif x1 == 0:

if y1 > 0:

if y2 == 0 and x2 > 0:

dem += 1

elif y1 < 0:

if y2 == 0 and x2 < 0:

dem += 1

print(dem)

Bài 6:

Viết hàm đảo mảng 2 chiều thành mảng đối xứng của nó theo trục ngang  
  
INPUT  
Dòng đầu tiên chứa hai số r và c là số dòng và số cột của mảng, mỗi số không quá 1000.  
  
r dòng tiếp theo, mỗi dòng chứa c số nguyên. Đây là các phần tử trong mảng 2 chiều  
  
OUTPUT  
Mảng sau khi đảo

r,c = map(int,input().split())

ab = [[0 for j in range(c)] for i in range(r)]

a = []

for i in range(0,r):

l = list(map(int,input().split()))

for j in range(0,c):

ab[i][j] = l[j]

for i in range(r-1,-1,-1):

for j in range(c-1,-1,-1):

print(ab[i][j], end= ' ')

print()

Bài 7:

2048 Là một tự game puzzle với luật chơi hết sức đơn giản nhưng khả năng gây nghiện rất cao đã làm mưa làm gió một thời trên khắp các ứng dụng từ nền tảng web cho đến di động vào năm 2014.  
  
Trò chơi 2048 bao gồm một bảng số kích thước 4 × 4 trong đó các ô số có thể trượt theo 4 hướng tương ứng với các phím mũi tên Up, Down, Left or Right. Khi người chơi xác định hướng trượt, những ô số này sẽ trượt xa hết mức có thể cho đến khi chúng đung phải một ô số khác hoặc đụng vào rìa của bảng số. Nếu hai ô số có giá trị cùng nhau va chạm với nhau khi trượt, chúng sẽ bị gộp thành một ô có giá trị gấp đôi, ô số được tạo thành nhờ cách gộp này sẽ không bị gộp thêm lần nữa trong cùng một lượt di chuyển.  
  
Ví dụ về sự thay đổi trạng thái bảng số khi người chơi chọn trượt sang hướng phím mũi tên Right.  
  
  
  
  
Ví dụ sự thay đổi của bảng số khi người dùng chọn trượt theo hướng mũi tênDown.  
  
  
Sinh viên thậm chí có thể chơi thử và xem mã nguồn của trò chơi tại địa chỉ: https://gabrielecirulli.github.io/2048/  
  
Yêu cầu hãy thử viết phần lõi của trò 2048. Đó là cho trạng thái hiện tại của bảng số và hướng trượt mà người chơi chọn, hãy tính ra trạng thái của bảng số sau khi kết thúc thao tác trượt.  
  
INPUT  
4 dòng đầu của input, mỗi dòng chứa 4 số nguyên cách nhau bởi khoảng trắng. Các số này tương ứng với các số trong bảng, những ô trống được ký hiệu bằng số 0 fasf  
  
Dòng cuối cùng của input chứa một trong 4 ký tự U, D, L, R. tượng trưng cho 1 trong 4 hướng Up, Down, Left, Right mà người dùng chọn  
  
OUTPUT  
Xuất ra 4 dòng, mỗi dòng chứa 4 số nguyên. Tương ứng với trạng thái của bảng sau khi trượt

ab = []

for i in range(4):

row = list(map(int, input().split()))

ab.append(row)

while True:

try:

move = input()

if move:

break

except:

Pass

def dichuyen(l):

k = [0]\*4

index = 0

for i in range(0,4):

if l[i] != 0:

if k[index] == 0:

k[index] = l[i]

elif k[index] == l[i] and l[i] != 2048:

k[index] \*= 2

index += 1

else:

index += 1

k[index] = l[i]

return k

r = [[0]\*4 for \_ in range(4)]

if move == 'L':

r = [dichuyen(m) for m in ab]

elif move == 'R':

r = [dichuyen(ab[i][::-1])[::-1] for i in range(4)]

elif move == 'U':

r = [dichuyen(col) for col in list(zip(\*ab))]

r = list(map(list, zip(\*r)))

elif move == 'D':

col = list(zip(\*ab[::-1]))

r = [dichuyen(col1)[::-1] for col1 in col]

r = list(map(list, zip(\*r)))

for i in range(0,4):

for j in range(0,4):

print(r[i][j], end= ' ')

print()

Bài 8:

Viết chương trình tìm dòng chứa số lớn nhất đầu tiên trong ma trận số nguyên (mxn) và xoá dòng đó đi. Các giá trị trong ma trận nằm trong khoảng [-32768, 32767], 0 < m, n < 1000.  
  
INPUT  
Gồm có mxn + 1 dòng:  
  
Dòng 1: Hai số m, n là số dòng và số cột của ma trận  
  
mxn dòng tiếp theo là các số trong ma trận  
  
OUTPUT  
Ma trận đã xoá đi dòng chứa số lớn nhất đầu tiên, mỗi dòng của ma trận xuất trên 1 hàng, các số trong 1 hàng cách nhau 1 khoảng trắng.

m,n = map(int,input().split())

a = []

for i in range(0,m\*n):

a.append(int(input()))

ab = [[0 for j in range(n)] for i in range(m)]

k = 0

index = (a.index(max(a))//n)

for i in range(0,m):

for j in range(0,n):

ab[i][j] = a[k]

k += 1

for i in range(0,m):

for j in range(0,n):

if i != index:

print(ab[i][j], end= ' ')

print()

Bài 9:

Làm mờ ảnh (bluring) là công việc thường dùng trong nhiếp ảnh (làm mờ nền cho đẹp) hay truyền hình (che thông tin nhạy cảm, đặc biệt là trong film nhật). Trong số các kỹ thuật làm mờ ảnh phổ biến thì đơn giản nhất là box blur. Kỹ thuật này xác định màu của một pixel là trung bình cộng màu của 9 pixel trong một ma trận 3x3 với tâm là pixel đang xét. Các pixel ở trên viền của bức ảnh thì bỏ đi không cần giữ.  
  
Biết rằng bức ảnh có thể được biểu diễn thành một ma trận các số nguyên. Mỗi phần tử trong ma trận tương ứng với một pixel và giá trị của phần tử chính là mã màu. Ban hãy làm mờ bức ảnh đó.  
  
INPUT  
Dòng đầu tiên chứa hai số r và c là số dòng và số cột của mảng.  
  
r dòng tiếp theo, mỗi dòng chứa c số nguyên. Đây là các phần tử trong mảng 2 chiều  
  
OUTPUT  
Ảnh sau khi box blur. Nếu ảnh không thể box blur thì không cần xuất gì cả

m,n = map(int,input().split())

ab = [list(map(int, input().split())) for \_ in range(m)]

for i, m in enumerate(ab[1:-1],1):

for j, n in enumerate(m[1:-1],1):

print((sum(ab[i-1][j-1:j+2]+ab[i][j-1:j+2]+ab[i+1][j-1:j+2]))//9, end= ' ')

print()

Bài 10:

Cho một mảng hai chiều các số nguyên. Hãy thực hiện dịch chuyển các phần tử ở rìa của mảng. Trong đó rìa thứ i sẽ dịch chuyển theo chiều kim đồng hồ nếu i là số chẵn và ngược chiều kim đồng hồ nếu i là số lẻ.  
  
Rìa thứ i của mảng được định nghĩa như sau:  
  
Rìa thứ 0 là hợp của dòng trên cùng, dòng cuối cùng, cột ngoài cùng bên trái và cột ngoài cùng bên phải.  
Nếu ta xóa rìa thứ 0 của mảng ban đầu ta sẽ thu được một mảng mới. Rìa thứ 0 của mảng mới chính là rìa thứ 1 của mảng ban đầu.  
Tương tự cho rìa thứ 2 và rìa thứ 3  
INPUT  
Dòng đầu tiên chứa hai số r và c là số dòng và số cột của mảng.  
  
r dòng tiếp theo, mỗi dòng chứa c số nguyên. Đây là các phần tử trong mảng 2 chiều  
  
OUTPUT  
Mảng sau khi dịch chuyển tất cả các rìa

m,n = map(int,input().split())

ab = [list(map(int, input().split())) for \_ in range(m)]

abr = [[0 for j in range(n)] for i in range(m)]

l = 0

i = 0

if n > 1 and m > 1:

while i < m\*n:

if l % 2 == 0:

if l == m // 2 and m < n:

for j in range(n - l - 1,l-1,-1):

abr[l][j] = ab[l][j-1]

i += 1

abr[l][l] = ab[l][n-l-1]

i += 1

elif l == n // 2 and m > n:

for j in range(l,m - l - 1):

abr[j][l] = ab[j+1][l]

i += 1

abr[m-l][l] = ab[l][l]

i += 1

else:

for j in range(l+1,n - l):

abr[l][j] = ab[l][j-1]

i += 1

for k in range(l+1,m - l):

abr[k][n - l - 1] = ab[k-1][n - l -1]

i += 1

for j in range(n - l - 2,-1+l,-1):

abr[m - l - 1][j] = ab[m - l - 1][j+1]

i += 1

for k in range(m - l - 2,-1+l,-1):

abr[k][l] = ab[k + 1][l]

i += 1

else:

if l == m // 2 and m < n:

for j in range(l,n - l - 1):

abr[l][j] = ab[l][j+1]

i += 1

abr[l][n-l-1] = ab[l][l]

i += 1

elif l == n // 2 and m > n:

for j in range(l,m - l - 1):

abr[j][l] = ab[j+1][l]

i += 1

abr[m-l-1][l] = ab[l][l]

i += 1

else:

for j in range(l,n - l - 1):

abr[l][j] = ab[l][j+1]

i += 1

for k in range(l+1,m - l):

abr[k][l] = ab[k - 1][l]

i += 1

for j in range(l+1,n - l):

abr[m - l - 1][j] = ab[m - l - 1][j-1]

i += 1

for k in range(m - l - 2,l-1,-1):

abr[k][n - l - 1] = ab[k+1][n - l -1]

i += 1

l += 1

if m\*n % 2 != 0 and i == m\*n - 1:

abr[m//2][n//2] = ab[m//2][n//2]

i += 1

for i in range(0,m):

for j in range(0,n):

print(abr[i][j], end= ' ')

print()

elif n == 1 and m > 1:

print(ab[m-1][0])

for i in range(0,m-1):

print(ab[i][0])

elif m == 1 and n > 1:

abr[0][0] = ab[0][n-1]

for i in range(0,n):

abr[0][i] = ab[0][i-1]

for j in range(0,n):

print(abr[0][j], end= ' ')

elif m == 1 and n == 1:

print(ab[0][0])

Bài 11:

Cho ma trận vuông gồm n dòng và n cột lưu trữ các số nguyên (0 < n ≤ 100). Kiểm tra ma trận trên có phải là ma trận đơn vị không?  
  
Định nghĩa: Ma trận đơn vị có số chiều n là một ma trận (nxn) trong đó mọi phần tử trên đường chéo chính bằng 1 và tất cả những phần tử khác đều bằng 0.  
  
INPUT  
Dãy các số nguyên trong đó: (Giả sử luôn thỏa điều kiện nhập)  
  
Số nguyên không âm đầu tiên: n dòng, n cột của ma trận  
Các số nguyên còn lại: giá trị của các phần tử của ma trận  
OUTPUT  
Xuất true nếu là ma trận đơn vị. Ngược lại xuất false

n = int(input())

ab = [[0 for j in range(n)] for i in range(n)]

h =[]

dem = 0

while True:

try:

l=input()

except EOFError:

break

for i in l.split():

if i!='':

h.append(int(i))

dem += 1

if int(i) != 0 and int(i) != 1:

print("false")

exit()

if dem != n\*n:

print("false")

exit()

else:

m = 0

for i in range(0,n):

for j in range(0,n):

ab[i][j] = h[m]

m += 1

def ktra(ab):

col = len(ab[0])

row = len(ab)

if col != row:

return False

if row == 0:

return False

for i in range(0,n):

for j in range(0,n):

if i == j and ab[i][j] != 1:

return False

elif i != j and ab[i][j] != 0:

return False

return True

if ktra(ab):

print("true")else:

print("false")

Bài 12:

Sau bao năm trời code thuê cày mướn, cắc củm dành dụm Bình cũng thuê được nhà. Ngày đầu tiên về sơn lại nhà mới Bình lại làm rơi nguyên thùng sơn xuống sàn. Sơn chảy ra nền nhà thành một vũng to trải trên nhiều viên gạch, Bình lại ngứa nghề tự hỏi không biết ranh giới của vùng gạch bị dính sơn là bao nhiêu? Bạn hãy viết chương trình tính giúp bình, xem mỗi cạnh ô gạch có độ dài 1 đơn vị  
  
INPUT  
Dòng đầu tiên chứa hai số tự nhiên m và n là kích thước sàn nhà của Bình (đơn vị là ô gạch)  
  
m dòng tiếp theo mỗi dòng chứa n số nguyên mang giá trị:  
  
Số 0 đại diện cho ô gạch chưa dính sơn  
Số 1 đại diện cho ô gạch đã bị dính sơn  
OUTPUT  
Độ dài ranh giới

m,n = map(int,input().split())

ab = [[0 for j in range(n+2)] for i in range(m+2)]

for i in range(1,m+1):

k = list(map(int,input().split()))

for j in range(1,n+1):

ab[i][j] = k[j-1]

dem = 0

for i in range(1,m+1):

for j in range(1,n+1):

if ab[i][j] == 1:

if ab[i+1][j] == 0:

dem += 1

if ab[i-1][j] == 0:

dem += 1

if ab[i][j+1] == 0:

dem += 1

if ab[i][j-1] == 0:

dem += 1

print(dem)

Bài 13:

Công ty TNHH văn hóa giải trí XYZ có một trang web bán sách điện tử. Trang web này bán hàng triệu đầu sách cả quốc văn lẫn ngoại văn. Do số lượng đầu sách quá lớn nên dù đã phân loại kỹ lưỡng và liệt kê đẹp mắt thì khách hàng của công ty vẫn gặp khó khăn khi muốn tìm một quyển sách ưng ý. Vì thế công ty quyết định trang bị cho trang web khả năng tìm kiếm, người dùng chỉ cần gõ vài dòng từ khóa vào ô tìm kiếm, trang web sẽ liệt kê tất cả đầu sách có tên chứa từ khóa người dùng nhập vào.  
  
Hãy viết chương trình giúp công ty XYZ cài đặt tính năng tìm kiếm nói trên.  
  
INPUT  
Phần đầu tiên của input là danh sách tên các đầu sách công ty đang bán. Mỗi tên đầu sách viết trên một dòng với độ dài không quá 250 ký tự. Phần này kết thúc bằng một dòng chỉ chứa dấu chấm câu “.” (Để đơn giản bài này giới hạn tối đa chỉ 10.000 đầu sách)  
  
Dòng cuối cùng của input là từ khóa người dùng đã nhập vào để tìm kiếm, từ khóa này không quá 200 ký tự  
  
OUTPUT  
Liệt kê tất cả tên những đầu sách có chứa từ khóa người dùng nhập vào. Mỗi tên xuất trên một dòng theo thứ tự xuất hiện của chúng trong input, vị trí từ khóa xuất hiện trong tên đầu sách được đánh dấu bằng cặp dấu ngoặc vuông [ và ]. Lưu ý có thể có những đầu sách trùng tên nhưng vẫn xem là các đầu sách khác nhau, lúc tìm kiếm không xét chữ hoa chữ thường nhưng lúc xuất kết quả phải đúng chữa hoa chữ thường.

books = []

while True:

book = input()

if book == '.':

break

books.append(book)

key = input()

thuongkey = key.lower()

kq = []

for book in books:

thuongbook = book.lower()

index = thuongbook.find(thuongkey)

if index != -1:

kq1 = book[:index] + '[' + book[index:index+len(key)] + ']' + book[index+len(key):]

print(kq1)