# Question 1

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Cho một dãy gồm n số nguyên. Viết chương trình:

* In ra các số chẵn theo thứ tự xuất hiện, nếu không có in ra “Khong co so chan”.
* Sau đó in ra các số lẻ theo thứ tự xuất hiện, nếu không có in ra “Khong co so le”.

Input:

* Dòng 1: Số nguyên n.
* Dòng 2: n số nguyên, các số cách nhau bởi dấu cách.

Output:

* Một dòng gồm các số chẵn hoặc thông báo “Khong co so chan”.
* Một dòng gồm các số lẻ hoặc thông báo “Khong co so le”.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 5  1 2 3 4 5 | 2 4  1 3 5 |
| 2 | 6  2 4 6 8 10 12 | 2 4 6 8 10 12  Khong co so le |

def phan\_loai(n, lst):

chan = [x for x in lst if x % 2 == 0]

le = [x for x in lst if x % 2 != 0]

if chan:

print(" ".join(map(str, chan)))

else:

print("Khong co so chan")

if le:

print(" ".join(map(str, le)))

else:

print("Khong co so le")

n = int(input())

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

phan\_loai(n, lst)

# Question 2

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Cho một số nguyên dương n và một số nguyên dương k. Viết chương trình tính tổng các số từ 1 đến n chia hết cho k.

**Input:**

* Dòng 1: Số nguyên n (1 ≤ n ≤ 10⁵).
* Dòng 2: Số nguyên k (1 ≤ k ≤ n).

**Output:**In ra tổng cần tìm.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 10  2 | 30 |

def totalt(n, k):

m = n // k

tong = k \* m \* (m + 1) // 2

return tong

n = int(input())

k = int(input())

print(totalt(n, k))

# Question 3

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Cho một dãy gồm **n số nguyên**. Kiểm tra xem dãy số có phải là **dãy tăng chặt** hay không. Biết dãy tăng chặt là dãy có mọi phần tử đứng sau **lớn hơn** phần tử đứng ngay trước nó.

**Input**

* Dòng 1: Số nguyên n (1 ≤ n ≤ 1000)
* Dòng 2: n số nguyên, cách nhau bởi dấu cách.

**Output**

* In ra YES nếu dãy tăng chặt
* In ra NO nếu ngược lại

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 5  1 2 3 4 5 | YES |
| 2 | 5  1 1 2 3 4 | NO |

def tang(n, lst):

for i in range(1, n):

if lst[i] <= lst[i-1]:

print("NO")

return

print("YES")

n = int(input())

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

tang(n, lst)

# Question 4

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Bàn cờ vua được biểu diễn dưới dạng hình vuông kích thước **n × n** ô. Để hỗ trợ người chơi dễ quan sát, các ô được tô màu **đen (B)** và **trắng (W)** xen kẽ nhau. Biết rằng ô đầu tiên ở góc trên bên trái luôn là**ô trắng (W)**.

**Yêu cầu:**

* Cho số nguyên n, hãy in ra bàn cờ vua kích thước n x n theo đúng quy tắc tô màu trên

**Input:**

* Là số nguyên n (1≤ n ≤100)

**Output:**

* In ra **n dòng**, mỗi dòng gồm **n ký tự**, mỗi ký tự là:
  + **'W'**: ô trắng
  + **'B'**: ô đen
* Các ô phải đảm bảo màu sắc xen kẽ theo quy luật bàn cờ vua.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 2 | 2 | WB  BW |
| 3 | 3 | WBW  BWB  WBW |

n = int(input())

for i in range(n):

row = ""

for j in range(n):

if (i + j) % 2 == 0:

row += 'W'

else:

row += 'B'

print(row)

# Question 5

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Để tính điểm rèn luyện cho sinh viên cuối học kì nhà trường quy định: Sinh viên phải tham gia các hoạt động để tích luỹ điểm, mỗi hoạt động có số điểm tích luỹ được quy định trước. Để có hạnh kiểm tốt, sinh viên phải có tổng điểm rèn luyện >= 70 điểm.

Viết chương trình nhập vào danh sách sinh viên và các hoạt động mà sinh viên đã tham gia. In ra danh sách các sinh viên chưa đủ điểm rèn luyện để đạt hạnh kiểm tốt cuối kì nhằm cảnh báo các bạn tích cực tham gia các hoạt động trong nhà trường.

**Input:**

+ Dòng đầu tiên nhập số nguyên dương n là số bộ dữ liệu về hoạt động của sinh viên (1<=n<=200).

+ n dòng tiếp theo, mỗi dòng nhập một bộ thông tin gồm: mã sinh viên, tên sinh viên, mã hoạt động. Các thông tin nhập vào đều kiểu chuỗi, không chứa dấu cách và cách nhau một khoảng trắng.

**Chú ý**: *Các sinh viên có thể tham gia một mã hoạt động nhiều lần (ở các thời điểm khác nhau).*

**Output:** Danh sách các sinh viên có **tổng điểm rèn luyện đã tích luỹ < 70** theo thứ tự: **mã sinh viên, tên sinh viên, tổng điểm rèn luyện**. Mỗi sinh viên trên một dòng. Nếu không có sinh viên nào không đủ điểm đạt hạnh kiểm tốt, in ra thông báo "**Khong co sinh vien nao diem ren luyen < 70**".

**Biết, danh sách các hoạt động và điểm rèn luyện tích luỹ trong kỳ cho trong bảng sau:**

|  |  |
| --- | --- |
| Mã HĐ | Điểm rèn luyện |
| hd1 | 5 |
| hd2 | 8 |
| hd2 | 10 |
| hd4 | 10 |
| hd5 | 15 |

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 2 | 6  SV01 An hd5  SV01 An hd5  SV01 An hd5  SV01 An hd5  SV01 An hd5  SV01 An hd5 | Khong co sinh vien nao diem ren luyen < 70 |

drl = {

'hd1': 5,

'hd2': 8,

'hd3': 10,

'hd4': 10,

'hd5': 15

}

def cal(n, data):

sv = []

for i in range(n):

masv, tensv, mahd = data[i]

diem = drl.get(mahd, 0)

res=False

for s in sv:

if s[0]== masv:

s\_idx=sv.index(s)

n\_s=s[2]+ diem

sv[s\_idx] =(masv,tensv,n\_s)

res=True

break

if not res:

sv.append((masv,tensv,diem))

svkdu = []

for masv,tensv,diem in sv:

if diem <70:

svkdu.append((masv, tensv, diem))

if svkdu:

for s in svkdu:

print(s[0], s[1], s[2])

else:

print("Khong co sinh vien nao diem ren luyen < 70")

n = int(input())

a = []

for \_ in range(n):

masv, tensv, mahd = input().split()

a.append((masv, tensv, mahd))

cal(n, a)

# Question 1

Partially correct

Marked out of 1.00

Flag question

Question text

Cho dãy số nguyên dương gồm N phần tử a1, a2​, ..., aN​. In ra số lượng các cặp (i,j) thỏa mãn:

* 1 ≤ i ≤ j ≤ N;
* K cho trước.

**Mô tả đầu vào:**

* Dòng đầu tiên chứa 2 số nguyên dương N và K.
* Dòng thứ hai gồm a1​,

**Ràng buộc:**

**Mô tả đầu ra:**

* Số lượng các cặp  thỏa mãn.

**Ví dụ:**

input:

3 5

1 2 1

ouput:

1

n,k=map(int,input().split())  
a=list(map(int,input().split()))  
fre = [0]\*501  
cnt=0  
for i in range(n):  
 res = k- a[i]\*a[i]  
 if 1 <= res <= 500:  
 cnt+=fre[res]  
 if a[i] + a[i]\*a[i] == k:  
 cnt+=1  
 fre[a[i]] +=1  
print(cnt)

# Question 2

Partially correct

Marked out of 1.00

Flag question

Question text

Một dãy số tự nhiên được gọi là "đẹp" nếu mỗi phần tử của nó lớn hơn tổng các phần tử đứng trước nó. Viết chương trình kiểm tra xem một dãy số tự nhiên có phải là dãy "đẹp" hay không.

**Input:**

* Dòng đầu tiên chứa số nguyên dương n (1 ≤ n ≤ 100), là số lượng phần tử của dãy.
* Dòng tiếp theo chứa n số nguyên dương là các phần tử của dãy, các phần tử cách nhau bởi 1 dấu cách.

**Output:**

* In ra YES hoặc NO tùy theo kết quả kiểm tra dãy đẹp hay không.

**Ví dụ:**

input:

4

1 3 7 15

ouput:

YES

# Question 3

Not complete

Marked out of 1.00

Flag question

Question text

Cho  số nguyên dương . Hãy ghép các số đó lại với nhau để được số nguyên dương lớn nhất.

Ví dụ: Với  số , ,  ta có  cách ghép được  số là: . Trong đó số lớn nhất là

**Input**

* Dòng đầu chứa số nguyên dương ;
* Dòng thứ hai chứa  số nguyên dương . Hai số liên tiếp cách nhau một dấu cách.

**Output**

* Ghi ra số nguyên dương lớn nhất ghép được.

**Constrainst**:

* .

.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  12 3 5 | 5312 |
| 3  12 907 91 | 9190712 |

from functools import cmp\_to\_key  
def cmp(a,b):  
 if a + b < b+ a :  
 return 1  
 elif a + b > b+ a:  
 return -1  
 else:  
 return 0  
n=int(input())  
a=input().split()  
a.sort(key=cmp\_to\_key(cmp))  
result = ''.join(a)  
print(result)

Question **4**

Correct

Mark 0.50 out of 1.00

Flag question

Question text

Dãy số tăng là dãy số mà trong đó mọi số đứng sau đều lớn hơn số đứng trước.

Viết hàm kiểm tra một mảng số thực gồm n phần tử có tăng hay ko? Hàm nhận 2 tham số là mảng a và số phần tử. Hàm trả về 0 nếu dãy a ko tăng, 1 nếu ngược lại.

Viết chương trình (hàm main) nhập mảng **a** gồm **n** số thực và mảng b gồm m số thực. Với **n, m** là các số nguyên dương nhập từ bàn phím.

Kiểm tra phần tử mảng a, b có phải là một dãy số tăng hay không?

**Input Format**

Dữ liệu vào nằm trên 4 dòng:

* Dòng 1: một số nguyên dương chỉ số phần từ mảng a.
* Dòng 2: các phần tử mảng a, số thực, cách nhau bởi dấu cách.
* Dòng 3: một số nguyên dương chỉ số phần từ mảng b.
* Dòng 4: các phần tử mảng b, số thực, cách nhau bởi dấu cách.

**Constraints**

* Mảng nhập vào có ít nhất 1 phần tử và nhiều nhất là 500 phần tử. (1<=n, m<=500)
* Kiểu các phần tử mảng là số thực

**Output Format**

**Hai dòng: thông báo kết quả của mảng a và b tương ứng**

* Nếu dãy số tăng, in ra **TANG**
* Ngược lại, in ra **KHONG\_TANG**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  2.00 4.50 6.12 8.30 10.99  5  2.00 4.50 6.15 10.95 8.00 | TANG  KHONG\_TANG |
| 5  2.00 4.50 6.15 10.95 8.00  5  2.00 4.50 6.12 8.30 10.99 | KHONG\_TANG  TANG |

def pl(n,a):

for i in range(1,n):

if a[i] <= a[i-1]:

print("KHONG\_TANG")

return

print("TANG")

n=int(input())

a=list(map(float,input().split()))

m=int(input())

b=list(map(float,input().split()))

pl(n,a)

pl(m,b)

Question **5**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tổng 2 ma trận đó và in kết quả ra màn hình. Nếu không tính được tổng 2 ma trận thì in ra thông báo "Du lieu vao sai"

**Input:**

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

**Output:**

+ Nếu tính được tổng:

- Dòng thứ nhất in ra thông báo "Ma tran tong"

- Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tổng:

- In ra thông báo "Du lieu vao sai"

**Constranins:**1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10 13  17 24 10 19 3 11 22 8 23 14 16 11 3  18 22 19 12 7 12 9 23 16 4 3 16 10  15 17 13 6 15 17 14 23 21 5 15 4 20  0 6 1 18 23 4 23 9 4 6 15 16 1  6 8 19 14 1 23 12 13 18 7 4 16 8  15 14 8 4 5 2 6 23 11 21 20 24 22  20 4 2 23 22 12 11 15 11 11 5 17 5  24 6 2 0 0 16 24 16 5 7 16 7 12  7 12 3 8 20 9 9 8 21 13 22 21 6  5 13 18 0 16 12 5 10 9 24 12 23 8  10 13  20 16 2 0 16 11 24 20 21 21 23 24 18  9 6 9 3 24 18 13 0 13 2 17 3 18  23 8 7 21 10 17 13 14 9 16 10 1 0  24 19 6 23 3 24 8 19 9 14 2 20 10  18 18 23 12 14 3 23 0 8 18 5 21 23  6 14 23 9 7 22 22 13 17 13 4 15 7  8 16 15 13 6 11 2 9 22 5 3 21 12  11 0 8 19 17 19 16 6 23 22 1 21 24  7 1 17 14 0 12 0 10 3 19 11 13 1  14 5 23 2 10 7 10 13 1 17 7 7 7 | Ma tran tong  37 40 12 19 19 22 46 28 44 35 39 35 21  27 28 28 15 31 30 22 23 29 6 20 19 28  38 25 20 27 25 34 27 37 30 21 25 5 20  24 25 7 41 26 28 31 28 13 20 17 36 11  24 26 42 26 15 26 35 13 26 25 9 37 31  21 28 31 13 12 24 28 36 28 34 24 39 29  28 20 17 36 28 23 13 24 33 16 8 38 17  35 6 10 19 17 35 40 22 28 29 17 28 36  14 13 20 22 20 21 9 18 24 32 33 34 7  19 18 41 2 26 19 15 23 10 41 19 30 15 |

def sumMatrix(a,b):

n=len(a)

m=len(a[0])

c=[]

for i in range(n):

row=[]

for j in range(m):

row.append(a[i][j]+b[i][j])

c.append(row)

return c

n1,m1=map(int,input().split())

a=[list(map(int,input().split())) for i in range(n1)]

n2,m2=map(int,input().split())

b=[list(map(int,input().split())) for i in range(n2)]

c=[]

if n1!=n2 or m1!=m2:

print("Du lieu vao sai")

else:

c=sumMatrix(a,b)

print("Ma tran tong")

for row in c:

print(\*row)

Question **7**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Việc bảo vệ máy tính để hạn chế người khác thâm nhập là một vấn đề đặt ra cho mọi nguời sử dụng máy tính. Để tăng tính an toàn trong lưu trữ, Bạn Nam đã quyết định giấu mật khẩu truy cập máy tính của mình vào một xâu S với một qui ước sao cho khi cần Nam có thể lấy lại được mật khẩu từ S.

Là một người yêu thích số học Nam chọn mật khẩu P là một số tự nhiên và mã hóa P=P1​×P2​×⋯×Pn​, với Pi​ là số nguyên dương và đem dấu các số Pi​ vào một xâu ký tự S.

Chẳng hạn xâu S = "mat22b3ma6c7d8" chứa mật khẩu là P=22×3×6×7×8.

**Yêu cầu**: Cho một xâu ký tự S, tìm mật khẩu P đã giấu trong xâu S.

Vì kết quả có thể rất lớn, in ra số dư của mật khẩu P khi chia cho .

**Mô tả đầu vào:**

Dữ liệu vào gồm 1 dòng duy nhất là xâu S, chỉ gồm chữ cái in thường và các chữ số.

**Mô tả đầu ra:**

Dữ liệu ra chứa một số duy nhât là kết quả của bài toán.

**Ví dụ:**

input:

mat22b3ma6c7d8

output:

22176

mod = 10\*\*9+7  
s=input().strip()  
res=1  
cur=0  
found =0  
for c in s:  
 if c.isdigit():  
 cur= cur\*10+int(c)  
 found=1  
 else:  
 if cur>0:  
 res = (res\*cur)%mod  
 cur=0  
if cur>0:  
 res=(res\*cur)%mod  
print(res)  
  
mod=10\*\*9+7  
s=input().strip()  
nums=[]  
tmp=""  
for c in s:  
 if c.isdigit():  
 tmp+=c  
 else:  
 tmp+=" "  
for x in tmp.split():  
 nums.append(int(x))  
res=1  
for i in nums:  
 res=(res\*i)%mod  
print(res)

Question **9**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Viết chương trình nhập vào danh sách gồm n hình tròn trong mặt phẳng. Nhập vào một điểm A (xa, ya) và điểm B (xb, yb). Cho biết điểm A, B nằm trong, nằm trên, hay nằm ngoài các hình tròn đã nhập vào. Biết thông tin của hình tròn gồm mã hình tròn, bán kính, tọa độ tâm (x, y).

Input:

+ Dòng thứ nhất nhập vào số nguyên dương n

+ Dòng tiếp theo nhập vào thông tin của từng hình tròn. Thứ tự nhập vào là mã, tâm x, tâm y, bán kính.

+ Dòng cuối cùng nhập vào xa, ya, xb, yb cách nhau dấu cách.

Output:

+ Gồm 2n dòng, mỗi dòng in ra thông báo

"Diem (xa, ya) nam trong/tren/ngoai hinh tron tam (x, y) ban kinh r"

"Diem (xb, yb) nam trong/tren/ngoai hinh tron tam (x, y) ban kinh r"

+ Trong đó: (xa, ya) và (xb, yb) là tọa độ điểm A và B, (x, y) và r là tọa độ tâm và bán kính các hình tròn trong danh sách.

Constrains:

+ 1<=n<=1000.

+ bán kính là số thực với độ chính xác 3 chữ số thập phân, tọa độ là các số nguyên.

+ Số pi =3.14159.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  88 17 57 9.000  32 69 54 11.000  21 76 29 30.000  68 25 55 31.000  34 41 12 16.667  45 60 18 53 | Diem (45, 60) nam ngoai hinh tron tam (17, 57) ban kinh 9.000  Diem (18, 53) nam trong hinh tron tam (17, 57) ban kinh 9.000  Diem (45, 60) nam ngoai hinh tron tam (69, 54) ban kinh 11.000  Diem (18, 53) nam ngoai hinh tron tam (69, 54) ban kinh 11.000  Diem (45, 60) nam ngoai hinh tron tam (76, 29) ban kinh 30.000  Diem (18, 53) nam ngoai hinh tron tam (76, 29) ban kinh 30.000  Diem (45, 60) nam trong hinh tron tam (25, 55) ban kinh 31.000  Diem (18, 53) nam trong hinh tron tam (25, 55) ban kinh 31.000  Diem (45, 60) nam ngoai hinh tron tam (41, 12) ban kinh 16.667  Diem (18, 53) nam ngoai hinh tron tam (41, 12) ban kinh 16.667 |

import math

def checkPoint(xa,ya,xb,yb,r):

d=math.sqrt((xa-xb)\*\*2+(ya-yb)\*\*2)

if d>r:

return "ngoai"

elif d< r :

return "trong"

else:

return "tren"

n=int(input())

res=[]

for i in range(n):

a=input().split()

ma=int(a[0])

pa=int(a[1])

pb=int(a[2])

r=float(a[3])

res.append((pa,pb,r))

xa,ya,xb,yb=map(int,input().split())

for pa,pb,r in res:

resA=checkPoint(xa,ya,pa,pb,r)

print("Diem ({}, {}) nam {} hinh tron tam ({}, {}) ban kinh {:.3f}".format(xa,ya,resA,pa,pb,r))

resb = checkPoint(xb,yb, pa, pb, r)

print("Diem ({}, {}) nam {} hinh tron tam ({}, {}) ban kinh {:.3f}".format(xb,yb, resb, pa,pb,r))

Question **10**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Viết chương trình nhập vào một danh sách gồm n nhân viên. Tìm nhân viên có lương thấp nhất. Nếu tìm được nhiều nhân viên cùng có lương thấp nhất thì in ra kết quả là nhân viên đầu tiên tìm được trong danh sách. Biết thông tin của một nhân viên gồm: tên, mã nv, hệ số lương, phụ cấp

Input:

+ Dòng thứ nhất nhập vào n là số nhân viên

+ Dòng tiếp theo nhập vào thông tin của các nhân viên theo thứ tự mã, tên, hệ số lương, phụ cấp.

Output:

+ Dòng thứ nhất in ra thông báo "Nhan vien co luong thap nhat"

+ Dòng thứ hai in ra thông tin của nhân viên tìm được. Các thông tin cách nhau dấu cách, thứ tự mã, tên, hệ số lương, phụ cấp, lương tháng in ở cuối mỗi dòng. Lương tháng được tính bằng hệ số lương \* 2000000 + phụ cấp.

Constrains: 1<=n<=200, mã nhân viên, phụ cấp là các số nguyên dương, tên nhân viên không chứa dấu cách, các thông tin còn lại là số thực có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  78 Anh 2.34 1000000  5 Vinh 3.30 500000  61 Nhung 2.67 700000  27 Trang 2.34 250000 | Nhan vien co luong thap nhat  27 Trang 2.34 250000 4930000.00 |

n=int(input())

res=[]

for i in range(n):

a=input().split()

ma=int(a[0])

ten=a[1]

hsl=float(a[2])

pc=int(a[3])

salary = hsl \* 2000000 + pc

res.append((ma,ten,hsl,pc,salary))

kq=min(res,key=lambda x:x[4])

print("Nhan vien co luong thap nhat")

print("{} {} {:.2f} {} {:.2f}".format(kq[0],kq[1],kq[2],kq[3],kq[4]))

Question **11**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Viết chương trình nhập vào một danh sách gồm n nhân viên. Tìm nhân viên có lương lớn nhất. Nếu có nhiều người có lương lớn nhất bằng nhau thì kết quả là người ở đầu danh sách. Biết thông tin của một nhân viên gồm: tên, mã nv, hệ số lương, phụ cấp

Input:

+ Dòng thứ nhất nhập vào n là số nhân viên

+ Dòng tiếp theo nhập vào thông tin của các nhân viên theo thứ tự tên, mã, hệ số lương, phụ cấp.

Output:

+ Dòng thứ nhất in ra thông báo "Nhan vien co luong lon nhat"

+ Dòng thứ hai in ra thông tin của nhân viên tìm được, các thông tin cách nhau dấu cách. Thứ tự mã nhân viên, tên nhân viên, hệ số lương, phụ cấp, lương tháng in ở cuối mỗi dòng. Lương tháng được tính bằng hệ số lương \* 2000000 + phụ cấp.

Constrains: 1<=n<=200, mã nhân viên, phụ cấp là các số nguyên dương, tên nhân viên không chứa dấu cách, các thông tin còn lại là số thực có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  Anh 78 2.34 1000000  Vinh 5 4.32 250000  Nhung 61 2.67 700000  Trang 27 4.32 250000 | Nhan vien co luong lon nhat  5 Vinh 4.32 250000 8890000.00 |

n=int(input())

kq=None

maxVal=-1

for i in range(n):

a=input().split()

ma=int(a[1])

ten=a[0]

hsl=float(a[2])

pc=int(a[3])

salary = hsl \* 2000000 + pc

if salary > maxVal:

maxVal=salary

kq=((ma,ten,hsl,pc,maxVal))

print("Nhan vien co luong lon nhat")

print("{} {} {:.2f} {} {:.2f}".format(kq[0],kq[1],kq[2],kq[3],kq[4]))

Công ty điện lực HP quản lý 3 loại khách hàng. Khách hàng là doanh nghiệp, khách hàng là hộ kinh doanh và khách hàng thông thường. Mỗi khách hàng được công ty lưu trữ với các thông tin sau: mã khách hàng, tên khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ.

+ tên  hàng không chứa dấu cách,

+ mã khách hàng là chuỗi ký tự, ký tự đầu là các chữ cái hoa D: cho doanh nghiệp, hoặc K: cho hộ kinh doanh, hoặc H: cho khách hàng thường.

Viết chương trình nhập vào một danh sách gồm n khách hàng.

In thông tin khách hàng là hộ kinh doanh phải trả nhiều tiền nhất. Nếu có nhiều khách hàng phải trả nhiều tiền nhất thì in ra khách hàng có số thứ tự đầu tiên tìm thấy. Nếu ko có khách hàng là hộ kinh doanh trong danh sách thì in ra thông báo "Khong co khach hang la ho kinh doanh."

Input:

+ Dòng thứ nhất nhập vào n là số khách hàng

+ Dòng tiếp theo nhập vào thông tin của các khách hàng theo thứ tự tên, mã số khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ.

Output:

+ In ra tên, mã khách, hàngchỉ số đầu kỳ, chỉ số cuối kỳ, lượng điện tiêu thụ, số tiền phải trả của khách hàng tìm được hoặc thông báo "Khong co khach hang la ho kinh doanh." Các thông tin in ra cách nhau dấu cách.

Constrains:

+ Lượng điện tiêu thụ = chỉ số cuối kỳ - chỉ số đầu kỳ.

+ Đơn giá là 1000 đồng/1kw cho khách hàng thông thường (với khách hàng là doanh nghiệp thì tăng đơn giá 5%, với khách hàng là hộ kinh doanh thì tăng đơn giá 7%):

+ 1<=n<=200,

+ các thông tin còn lại là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  Yen H919 255 363  Trang K99 43 643  Yen H9191 63 363  Thu D72 5 82  Trang K919 431 643 | Trang K99 43 643 600 642000 |

n=int(input())

kq=None

maxVal=-1

maxVal=-1e9

for i in range(n):

a=input().split()

ma=a[1]

ten=a[0]

start=int(a[2])

end=int(a[3])

consume= end-start

price=0

if ma[0] == 'K':

price=consume\*1070

if price>maxVal:

maxVal=price

kq=(ten,ma,start,end,consume,maxVal)

if kq is None:

print("Khong co khach hang la ho kinh doanh.")

else:

print(\*kq)

Question **13**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Viết chương trình nhập vào một danh sách gồm n thí sinh thi đại học. In thông tin thí sinh thủ khoa (thi đỗ và có tổng điểm cao nhất) ra màn hình.

   +Nếu có nhiều thủ khoa thì in ra thủ khoa có số thứ tự nhỏ nhất trong danh sách sách.

   + Nếu không có thủ khoa (mọi thí sinh đều trượt) thì thông báo "Khong co thi sinh thi do"

   + Biết thông tin của một thí sinh gồm: tên, số báo danh, điểm môn 1, điểm môn 2, điểm môn 3.

   + Thí sinh thi đỗ khi tổng điểm >= 15.00 và điểm các môn >=1.00.

Input:

+ Dòng thứ nhất nhập vào n là số thí sinh

+ Dòng tiếp theo nhập vào thông tin của các thí sinh theo thứ tự tên, số báo danh, điểm môn 1, điểm môn 2, điểm môn 3.

Output:

Nếu tìm được thủ khoa:

+ Dòng thứ nhất in ra thông báo "So thu tu cua thu khoa: X", X là số thự tự của thí sinh đỗ thủ khoa tìm được theo yêu cầu.

+ Dòng thứ hai in ra thông tin của thí sinh thủ khoa tìm được. Các thông tin cách nhau dấu cách (thứ tự: số báo danh, tên, điểm môn 1, điểm môn 2, điểm môn 3), tổng điểm in ở cuối mỗi dòng.

Nếu KHÔNG tìm được thủ khoa:

+ In ra thông báo "Khong co thi sinh thi do"

Constrains: 1<=n<=200, điểm các môn và tổng điểm có độ chính xác 2 chữ số thập phân, số báo danh là số nguyên dương, tên thí sinh không chứa dấu cách.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  Huy 87 5.00 3.00 3.00  Trang 76 9.00 9.00 9.00  Lan 2 3.00 2.00 3.00  Anh 25 8.00 9.00 10.00  Van 35 0.00 10.00 9.00 | So thu tu cua thu khoa: 1  76 Trang 9.00 9.00 9.00 27.00 |

n=int(input())

idx=-1

kq=None

maxVal=-1e9

for i in range(n):

a=input().split()

ten=a[0]

ma=int(a[1])

d1=float(a[2])

d2=float(a[3])

d3=float(a[4])

tong=d1+d2+d3

if tong>=15.00 and d1>=1.00 and d2>=1.00 and d3>=1.00:

if tong>maxVal:

maxVal=tong

idx=i

kq=(ma,ten,d1,d2,d3,tong)

if kq is None:

print("Khong co thi sinh thi do")

else:

print("So thu tu cua thu khoa:",idx)

print("{} {} {:.2f} {:.2f} {:.2f} {:.2f}".format(\*kq))

Question **14**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Công ty Điện lực HP quản lý danh sách n khách hàng với các thông tin: mã khách hàng, tên khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ. Viết chương trình nhập vào một danh sách gồm n khách hàng. In thông tin khách hàng phải trả nhiều tiền nhất. Nếu có nhiều khách hàng phải trả nhiều tiền nhất thì in ra khách hàng có số thứ tự đầu tiên tìm thấy.

Input:

+ Dòng thứ nhất nhập vào n là số khách hàng

+ Dòng tiếp theo nhập vào thông tin của các khách hàng theo thứ tự tên, mã số khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ.

Output:

+ Dòng thứ nhất in ra thông báo: "Khach hang phai tra tien nhieu nhat: X". Với X là số thứ tự của khách hàng tìm được.

+ Dòng tiếp theo in ra thông tin của khách hàng tìm được. Các thông tin cách nhau dấu cách (thứ tự: mã khách hàng, tên, chỉ số đầu kỳ, chỉ số cuối kỳ, lượng điện tiêu thụ, số tiền phải trả.

+ Lượng điện tiêu thụ = chỉ số cuối kỳ - chỉ số đầu kỳ.

+ Số tiền phải trả được tính như sau:

- 100 số (kW) đầu tiên đơn giá là 1000 đồng.

- 100 số (kW) tiếp theo đơn giá là 1500 đồng.

- Từ 201số (kW) trở lên thì đơn giá là 2000 đồng.

Constrains: 1<=n<=200, tên thí sinh không chứa dấu cách, các thông tin còn lại là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  Yen 919 55 363  Trang 99 10 643  Thu 5 72 82 | Khach hang phai tra tien nhieu nhat: 1  99 Trang 10 643 633 1116000 |

n=int(input())

res=None

idx=-1

maxVal=-1e9

for i in range(n):

a=input().split()

ten=a[0]

ma=int(a[1])

start=int(a[2])

end=int(a[3])

consume = end-start

price=0

if consume < 100:

price=consume\*1000

elif 100<consume<200:

price= 1000\*100 + (consume-100)\*1500

else:

price= 1000\*100 + 100\*1500 + (consume-200)\*2000

if price>maxVal:

maxVal=price

idx=i

res=(ma,ten,start,end,consume,maxVal)

print("Khach hang phai tra tien nhieu nhat:",idx)

print(\*res)

# Question 5

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tổng 2 ma trận đó và in kết quả ra màn hình. Nếu không tính được tổng 2 ma trận thì in ra thông báo "Du lieu vao sai"

**Input:**

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

**Output:**

+ Nếu tính được tổng:

- Dòng thứ nhất in ra thông báo "Ma tran tong"

- Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tổng:

- In ra thông báo "Du lieu vao sai"

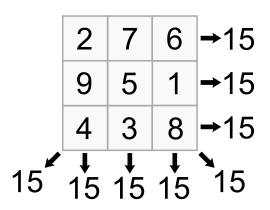
**Constranins:**1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10 13  17 24 10 19 3 11 22 8 23 14 16 11 3  18 22 19 12 7 12 9 23 16 4 3 16 10  15 17 13 6 15 17 14 23 21 5 15 4 20  0 6 1 18 23 4 23 9 4 6 15 16 1  6 8 19 14 1 23 12 13 18 7 4 16 8  15 14 8 4 5 2 6 23 11 21 20 24 22  20 4 2 23 22 12 11 15 11 11 5 17 5  24 6 2 0 0 16 24 16 5 7 16 7 12  7 12 3 8 20 9 9 8 21 13 22 21 6  5 13 18 0 16 12 5 10 9 24 12 23 8  10 13  20 16 2 0 16 11 24 20 21 21 23 24 18  9 6 9 3 24 18 13 0 13 2 17 3 18  23 8 7 21 10 17 13 14 9 16 10 1 0  24 19 6 23 3 24 8 19 9 14 2 20 10  18 18 23 12 14 3 23 0 8 18 5 21 23  6 14 23 9 7 22 22 13 17 13 4 15 7  8 16 15 13 6 11 2 9 22 5 3 21 12  11 0 8 19 17 19 16 6 23 22 1 21 24  7 1 17 14 0 12 0 10 3 19 11 13 1  14 5 23 2 10 7 10 13 1 17 7 7 7 | Ma tran tong  37 40 12 19 19 22 46 28 44 35 39 35 21  27 28 28 15 31 30 22 23 29 6 20 19 28  38 25 20 27 25 34 27 37 30 21 25 5 20  24 25 7 41 26 28 31 28 13 20 17 36 11  24 26 42 26 15 26 35 13 26 25 9 37 31  21 28 31 13 12 24 28 36 28 34 24 39 29  28 20 17 36 28 23 13 24 33 16 8 38 17  35 6 10 19 17 35 40 22 28 29 17 28 36  14 13 20 22 20 21 9 18 24 32 33 34 7  19 18 41 2 26 19 15 23 10 41 19 30 15 |

# Bài 1:

Trong toán vui, một ma trận kì ảo bậc **n** (còn gọi là ma phương hay hình vuông ma thuật) là một cách sắp xếp **n²** số, thường là các số nguyên phân biệt, trong một bảng vuông sao cho tổng **n** số trên mỗi hàng, cột, và đường chéo đều bằng nhau. Ma trận kì ảo chuẩn chứa các số nguyên từ 1 đến **n²**. (Wikipedia)



Viết chương trình nhập vào 2 ma trận X và Y có kích thước tương ứng là nxn và mxm. Kiểm tra 2 ma trận X, Y có phải ma trận kì ảo chuẩn không.

**Input Format**

Dữ liệu nhập vào trên **n+1** dòng (cho ma trận X):

* Dòng 1: bậc của ma trận – **n**.
* **n** dòng còn lại: các dòng của ma trận, mỗi dòng gồm **n** số nguyên, cách nhau bởi dấu cách.

Tiếp theo:

Dữ liệu nhập vào trên m**+1** dòng (cho ma trận Y):

* + Dòng 1: bậc của ma trận – **m**.
  + **m** dòng còn lại: các dòng của ma trận, mỗi dòng gồm **m** số nguyên, cách nhau bởi dấu cách.

**Constraints**

* 1 <= **n, m** <= 100

**Output Format**

**In ra 2 dòng, ứng với kết quả ma trận X, Y tương ứng**

* Nếu là ma trận kì ảo, in ra **MAGIC SQUARE**
* Ngược lại, in ra **NOT MAGIC**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  4 9 2  3 5 7  8 1 6  4  2 16 13 3  11 5 8 10  7 9 12 6  14 4 1 15 | MAGIC SQUARE  MAGIC SQUARE |

Giải:

def is\_magic\_square(matrix):

n = len(matrix)

target\_sum = sum(matrix[0])

for row in matrix:

if sum(row) != target\_sum:

return False

for col in range(n):

if sum(matrix[row][col] for row in range(n)) != target\_sum:

return False

if sum(matrix[i][i] for i in range(n)) != target\_sum:

return False

if sum(matrix[i][n - 1 - i] for i in range(n)) != target\_sum:

return False

return True

n = int(input())

matrix\_X = [list(map(int, input().split())) for \_ in range(n)]

m = int(input())

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

if is\_magic\_square(matrix\_X):

print("MAGIC SQUARE")

else:

print("NOT MAGIC")

if is\_magic\_square(matrix\_Y):

print("MAGIC SQUARE")

else:

print("NOT MAGIC")

Bài 2:

Viết chương trình thực hiện:

1. Nhận vào ma trận gồm các giá trị **0** và **1** biểu diễn 1 ảnh bitmap trắng đen (1 – đen, 0 – trắng).
2. Dùng kí tự trong bảng mã ASCII để hiển thị minh họa ảnh bitmap ra màn hình.

**Input Format**

Dữ liệu nhập vào trên **n+1** dòng:

* Dòng 1: số hàng (**n**) và số cột (**m**) của ma trận.
* **n** dòng còn lại: các dòng của ma trận, mỗi dòng gồm **m** giá trị **0**/**1**, cách nhau bởi dấu cách.

**Constraints**

* 1 <= **n, m** <= 100

**Output Format**

* In ảnh minh họa trên **n** dòng.
* Trên mỗi dòng, in ra các kí tự trắng hoặc kí tự \* tương ứng với giá trị **0** hoặc **1**. Các kí tự cách nhau 1 dấu cách.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 8 8  0 0 1 1 1 1 0 0  0 1 0 0 0 0 1 0  1 0 1 0 0 1 0 1  1 0 0 0 0 0 0 1  1 0 1 0 0 1 0 1  1 0 0 1 1 0 0 1  0 1 0 0 0 0 1 0  0 0 1 1 1 1 0 0 | \* \* \* \*  \* \*  \* \* \* \*  \* \*  \* \* \* \*  \* \* \* \*  \* \*  \* \* \* \* |

Giải:

def print\_bitmap(n, m, matrix):

for i in range(n):

for j in range(m):

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

print('\*', end=' ')

else:

print(' ', end=' ')

print()

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

matrix = []

for \_ in range(n):

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

matrix.append(row)

print\_bitmap(n, m, matrix)

Bài 3:

Cho hai số m, n. Hãy loại bỏ một chữ số của m để tạo thành một số p nhỏ nhất chia hết cho n.

* Input: hai số nguyên dương m, n cách hau một dấu cách
* Output: số p tìm được, nếu không tìm được in ra "impossible"
* Ràng buộc:
  + 10 < m < 1000000
  + 1 < n < 100000

**For example:**

| **Input** | **Result** |
| --- | --- |
| 248 12 | 24 |
| 208 4 | 8 |
| 229 23 | impossible |

Giải:

def remove\_digit(m, n):

m\_str = str(m)

return min([int(m\_str[:i] + m\_str[i+1:]) for i in range(len(m\_str)) if int(m\_str[:i] + m\_str[i+1:]) % n == 0], default="impossible")

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

print(remove\_digit(m, n))

Bài 4:

Viết hàm chuẩn hóa một chuỗi sao cho chữ cái đầu chuỗi hoặc sau dấu chấm (.) viết hoa, các chữ cái còn lại viết thường, sau dấu chấm, dấu phẩy có đúng một dấu cách.

Viết chương trình nhập vào hai chuỗi gồm các chữ cái, chữ số, dấu chấm, dấu phẩn và dấu cách. Sử dụng hàm chuẩn hóa ở trên với 2 chuỗi vừa nhập.

Input: Dòng thứ nhất nhập chuỗi X; dòng thứ 2 nhập chuỗi Y

Output: Dòng thứ nhất in ra chuỗi kết quả X; dòng thứ hai in ra chuỗi kết quả Y

Constrains: chuỗi ký tự chỉ chứa các chữ cái, chữ số, dấu chấm, dấu phẩy và dấu cách.

Examples:

X: "chao cac ban.toi dang hoc bai."

Y: "em thich hoa hong,hoa lan,       hoa dao"

Kết quả:

X: "Chao cac ban. Toi dang hoc bai. "

Y: "Em thich hoa hong, hoa lan, hoa dao"

**For example:**

| **Input** | **Result** |
| --- | --- |
| chao cac ban.toi dang hoc bai.  em thich hoa hong,hoa lan, hoa dao | Chao cac ban. Toi dang hoc bai.  Em thich hoa hong, hoa lan, hoa dao |

Giải:

import re

def normalize\_string(s):

s = s.lower()

s = re.sub(r'\s\*\.\s\*', '. ', s)

s = re.sub(r'\s\*,\s\*', ', ', s)

s = '. '.join([i.capitalize() for i in s.split('. ')])

return s

X = input().strip()

Y = input().strip()

print(normalize\_string(X))

print(normalize\_string(Y))

Bài 5:

Trong mật mã học, Caesar là hệ mã cổ điển được đặt theo tên của một vĩ nhân trong lịch sử La Mã - Gaius Julius Caesar. Hệ mã này sử dụng một số tự nhiên **k** làm khoá để mã hoá hoặc giải mã văn bản. Việc mã hoá hay giải mã được thực hiện bằng cách dịch chuyển vị trí các kí tự trong bảng chữ cái đi **k** vị trí. Ví dụ với bảng chữ cái **A gồm** 10 kí tự {'a', 'n', 'o', 'b', ' ', 't', 'i', 'm', 'e', 'd'}**,** và khoá **k** = 2:

* Việc mã hoá được thực hiện bằng cách dịch các chữ cái sang phải 2 vị trí. Kí tự n được mã hoá thành b, e được mã hoá thành a. Như vậy bản rõ '**no time to die**' sẽ được mã hoá thành '**b imedaim inea**'.
* Việc giải mã được thực hiện bằng cách dịch các chữ cái sang trái 2 vị trí. Kí tự b được giải mã thành n, a được giải mã thành e. Như vậy bản mã '**b imedaim inea**' sẽ được giải mã thành '**no time to die**'.

Hãy viết chương trình mã hoá và giải mã sử dụng hệ Caesar theo mô tả trên.

* Đầu vào:
  + Bảng chữ cái **A** gồm **n** kí tự viết liền nhau
  + Khoá **k**
  + Một xâu kí tự **s**
  + Một số nhị phân thể hiện yêu cầu cần thực hiện trên xâu **s**, với 0 là mã hoá, 1 là giải mã.
* Đầu ra: xâu kết quả của việc mã hoá / giải mã.
* Ràng buộc: 0 < **k** ≤≤ **n**

**For example:**

| **Input** | **Result** |
| --- | --- |
| anob timed  2  no time to die  0 | b imedaim inea |
| anob timed  2  b imedaim inea  1 | no time to die |

Giải:

def caesar\_cipher(char\_set, key, s, operation):

n = len(char\_set)

result = ""

for char in s:

if char in char\_set:

index = (char\_set.index(char) + key) % n if operation == 0 else (char\_set.index(char) - key) % n

result += char\_set[index]

else:

result += char

return result

char\_set = input().strip()

key = int(input())

s = input().strip()

operation = int(input())

result = caesar\_cipher(char\_set, key, s, operation)

print(result)

Bài 6:

Phần tử đặc biệt trong một ma trận là phần tử xuất hiện trên tất cả các hàng và tất cả các cột. Hãy viết chương trình tìm các phần tử đặc biệt trong một ma trận.

* Đầu vào:
  + Dòng đầu tiên gồm hai số n, m cách nhau bởi dấu cách là số hàng và số cột của ma trận
  + n dòng tiếp theo, mỗi dòng chứa m số là các phần tử trong một hàng, cách nhau bởi dấu cách
* Đầu ra: các phần tử đặc biệt tìm được, cách nhau bởi dấu cách
* Ràng buộc: n, m là các số nguyên dương.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4  0 1 2 3  1 2 4 1  2 6 1 2 | 1 2 |

Giải:

def find\_special\_elements(matrix):

n, m = len(matrix), len(matrix[0])

unique\_elements = set(matrix[0])

for i in range(1, n):

unique\_elements.intersection\_update(matrix[i])

return list(unique\_elements)

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

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

result = find\_special\_elements(matrix)

print(\*result)

Bài 7:

Một câu được gọi là đối xứng nếu sau khi loại bỏ các dấu cách và các kí tự đặc biệt (chỉ giữ lại chữ thường, chữ hoa và chữ số), rồi chuyển các chữ cái thành chữ thường thì viết theo thứ tự ngược lại vẫn được câu đó. Ví dụ các câu sau được gọi là các câu đối xứng:

* Stressed desserts!
* Now I see bees, I won.

Hãy viết chương trình kiểm tra xem một câu có phải là câu đối xứng hay không?

* Đầu vào: câu s1 nằm trên dòng thứ nhất, câu s2 nằm trên dòng thứ hai
* Đầu ra: 'true' nếu s1, s2 là câu đối xứng, 'false' nếu ngược lại. Kết quả được in trên 2 dòng khác nhau.

**For example:**

| **Input** | **Result** |
| --- | --- |
| Now I see bees, I won.  I tried my best to pass the exam! | true  false |
| I tried my best to pass the exam!  I tried my best to pass the exam! | false  false |

Giải:

def is\_palindrome(sentence):

clean\_sentence = ''.join(c.lower() for c in sentence if c.isalnum())

return clean\_sentence == clean\_sentence[::-1]

s1 = input().strip()

s2 = input().strip()

result\_s1 = 'true' if is\_palindrome(s1) else 'false'

result\_s2 = 'true' if is\_palindrome(s2) else 'false'

print(result\_s1)

print(result\_s2)

Bài 8:  
Một xâu kí tự được gọi là đối xứng nếu viết theo chiều ngược lại ta vẫn được xâu đó. Cho một xâu kí tự, hãy tìm xâu con đối xứng dài nhất.

* Đầu vào: xâu s
* Đầu ra: xâu con đối xứng dài nhất của s, nếu có nhiều xâu như vậy, hãy in ra xâu xuất hiện đầu tiên theo thứ tự từ trái sang phải.
* Ràng buộc: xâu s có ít nhất một kí tự

**For example:**

| **Input** | **Result** |
| --- | --- |
| shissi white | issi |
| easy | e |

Giải:

def longest\_palindromic\_substring(s):

n = len(s)

start, max\_len = 0, 1

for i in range(1, n):

odd\_len = expand\_around\_center(s, i, i)

even\_len = expand\_around\_center(s, i - 1, i)

curr\_len = max(odd\_len, even\_len)

if curr\_len > max\_len:

max\_len = curr\_len

start = i - curr\_len // 2

return s[start:start + max\_len]

def expand\_around\_center(s, left, right):

n = len(s)

while left >= 0 and right < n and s[left] == s[right]:

left -= 1

right += 1

return right - left - 1

s = input()

result = longest\_palindromic\_substring(s)

print(result)

Bài 9:

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số nguyên. Sắp xếp từng hàng của ma trận tăng dần. In kết quả ra màn hình.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output:

- Dòng thứ nhất in ra n và m cách nhau dấu cách

- Dòng tiếp theo in ra ma trận kết quả

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 6  17 23 9 3 0 9  12 2 21 12 23 7  12 2 2 13 7 6  7 24 22 12 11 20  4 5 7 24 2 11  15 23 20 22 20 24 | 6 6  0 3 9 9 17 23  2 7 12 12 21 23  2 2 6 7 12 13  7 11 12 20 22 24  2 4 5 7 11 24  15 20 20 22 23 24 |

Giải:

def sort\_matrix\_rows(matrix):

sorted\_matrix = [sorted(row) for row in matrix]

return sorted\_matrix

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

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

sorted\_matrix = sort\_matrix\_rows(matrix)

print(n,m)

for row in sorted\_matrix:

print(\*row)

Bài 10:

Viết chương trình nhập vào một mảng các số nguyên. Sắp xếp mảng đó tăng dần.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng (đã được sắp xếp tăng dần) cách nhau dấu cách.

Constrains:

+ Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  169 224 228 108 212 214 205 145 77 211 241 | 11  77 108 145 169 205 211 212 214 224 228 241 |

Giải:

N = int(input())

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

arr.sort()

print(N)

print(\*arr)

Bài 11:

Cho một dãy số tự nhiên a và một số nguyên b. Viết chương trình tìm số lớn nhất chia hết cho b.

* Input: Mỗi test bao gồm hai dòng. Dòng đầu tiên là dãy a, các phần tử cách nhau bởi một khoảng trắng. Dòng thứ hai chứa số b.
* Output: Số lớn nhất chia hết cho b, hoặc 0 (nếu không tìm thấy số nào chia hết cho b)

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 3 8 4 9 2 10  2 | 10 |

Giải:

def find\_largest\_multiple(a, b):

a.sort(reverse=True) # Sắp xếp dãy số giảm dần

for num in a:

if num % b == 0:

return num

return 0

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

b = int(input())

result = find\_largest\_multiple(a, b)

print(result)

Bài 12:

Cho n khối hộp chữ nhật, hãy tìm khối hộp có thể tích lớn nhất.

* Đầu vào:
  + Dòng đầu tiên là số n
  + n dòng tiếp theo chứa thông tin của n khối hộp chữ nhật, mỗi dòng gồm ba số cách nhau bởi dấu cách là các số đo chiều dài, chiều rộng, và chiều cao của khối hộp.
* Đầu ra: bốn số cách nhau bởi dấu cách là số đo chiều dài, chiều rộng, chiều cao, và thể tích của khối hộp chữ nhật có thể tích lớn nhất. Kết quả hiển thị với độ chính xác 1 chữ số sau dấu phẩy.
* Ràng buộc:
  + n là số nguyên dương.
  + Số đo các chiều của một khối hộp là số thực dương.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  2.0 6.4 5.0  2.6 3.2 8.0  3.1 1.1 6.0 | 2.6 3.2 8.0 66.6 |

Giải:

def find\_max\_volume\_box(n, boxes):

max\_volume = 0

max\_box = []

for box in boxes:

length, width, height = box

volume = length \* width \* height

if volume > max\_volume:

max\_volume = volume

max\_box = box

return max\_box + [round(max\_volume, 1)]

n = int(input())

boxes = []

for i in range(n):

box = list(map(float, input().split()))

boxes.append(box)

result = find\_max\_volume\_box(n, boxes)

print(\*result)

Bài 13:

Viết hàm nhận đầu vào là số nguyên h khác 0. In ra tam giác cân với chiều cao |h| bằng các ký tự \* theo quy tắc sau. Nếu h>0 thì đáy của tam giác ở phía dưới. Nếu h<0 thì đáy của tam giác ở phía trên.

Ví dụ: h=-5 thì tam giác cần vẽ là

 \*\*\*\*\*\*\*\*\*

  \*\*\*\*\*\*\*

   \*\*\*\*\*

    \*\*\*

     \*

Với h=5 thì tam giác cần vẽ là

    \*

   \*\*\*

  \*\*\*\*\*

 \*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

Viết chương trình nhập vào 3 số nguyên x, y, z. Dùng hàm trên để vẽ các tam giác với chiều cao tương ứng với |x|, |y|, |z|.

Input: 3 số nguyên x, y, z cách nhau dấu cách.

Output:

Dòng 1 in số nguyên x

Từ dòng 2 in ra tam giác với chiều cao |x|

Tiếp theo đến y, z

Constrains: các dữ liệu là kiểu số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 -8 1 | 6  \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*  -8  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*  \*\*\*  \*  1  \* |

Giải:

def Xuoi(n):

for i in range(0,n):

for j in range(0,n-i-1):

print(" ", end="")

for j in range(0,2\*i+1):

print("\*", end="")

print()

def Nguoc(n):

for i in range(0,n):

for j in range(0,i):

print(" ", end="")

for j in range(0,2\*n-2\*i-1):

print("\*", end="")

print()

x,y,z=map(int,input().split())

print(x)

if x>0: Xuoi(x)

else:

x=-x

Nguoc(x)

print(y)

if y>0: Xuoi(y)

else:

y=-y

Nguoc(y)

print(z)

if z>0: Xuoi(z)

else:

z=-z

Nguoc(z)

Bài 14:

Viết hàm đổi số nguyên dương n từ thập phân sang hệ nhị phân. Chuỗi nhị phân nằm trên cùng một dòng, cách nhau dấu cách. Ví dụ: 1 0 0 1

Viết chương trình nhập hai số nguyên dương a, b từ bàn phím. Sử dụng hàm trên để đổi số các số nguyên dương từ a đến b sang hệ nhị phân. Trong trường hợp b <= a thì đổi các số nguyên dương từ b đến a sang hệ nhị phân.

Input: a, b trên cùng một dòng, cách nhau dấu cách.

Output:  Mỗi dòng thể hiện như sau (ví dụ với a=93, b=95)

93: 1 0 1 1 1 0 1

94: 1 0 1 1 1 1 0

95: 1 0 1 1 1 1 1

Constrains: các số kiểu int

**For example:**

| **Input** | **Result** |
| --- | --- |
| 104 92 | 92: 1 0 1 1 1 0 0  93: 1 0 1 1 1 0 1  94: 1 0 1 1 1 1 0  95: 1 0 1 1 1 1 1  96: 1 1 0 0 0 0 0  97: 1 1 0 0 0 0 1  98: 1 1 0 0 0 1 0  99: 1 1 0 0 0 1 1  100: 1 1 0 0 1 0 0  101: 1 1 0 0 1 0 1  102: 1 1 0 0 1 1 0  103: 1 1 0 0 1 1 1  104: 1 1 0 1 0 0 0 |

Giải:

def decimal\_to\_binary(n):

if n == 0:

return "0"

binary=""

while n > 0:

remainder = n % 2

binary= str(remainder) + binary

n //= 2

return binary

def convert\_decimal\_range\_to\_binary(a, b):

start, end = min(a, b), max(a, b)

for i in range(start, end + 1):

binary\_num = decimal\_to\_binary(i)

print("{}: {}".format(i, ' '.join(binary\_num)))

a, b = map(int, input().split())

convert\_decimal\_range\_to\_binary(a, b)

Bài 15:

**Bạn hãy viết hàm có đầu vào là số nguyên n. Nếu n là số dương thì liệt kê các chữ số lẻ của n theo chiều từ trái qua phải. Các chữ số cách nhau dấu cách. Nếu không có chữ số lẻ nào thì in ra chữ "No". Nếu dữ liệu vào (n) là số âm thì in ra chữ "Error".**

**Viết chương trình nhập vào ba số nguyên x, y, z. Thực hiện hàm trên với các số x, y, z. Kết quả in trên 3 dòng.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output:**

**1 9**

**Error**

**No**

**Example 2:**

**+ Input: 190 172 115**

**+ Output:**

**1 9**

**1 7  
1 1 5**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | 1 9  Error  No |

Giải:

def kq(n):

dem = 0

a = []

if n < 0:

print("Error")

else:

while n != 0:

if (n % 10) % 2 != 0:

a.append(n % 10)

dem += 1

n //= 10

if dem == 0:

print("No")

else:

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

print(a[i], end=" ")

print()

x, y, z = map(int, input().split())

kq(x)

kq(y)

kq(z)

Bài 16:

Cho một ma trận, hãy sắp xếp các phần tử trong từng hàng theo thứ tự tăng, giảm xen kẽ theo thứ tự từ trái qua phải. Ví dụ, nếu hàng thứ nhất sắp xếp theo chiều tăng dần thì hàng thứ hai sắp xếp theo chiều giảm dần.

* Đầu vào:
  + Dòng đầu tiên là hai số m, n đại diện cho số hàng và số cột của ma trận, cách nhau bởi dấu cách
  + m dòng tiếp theo, mỗi dòng gồm n số là các phần tử trên một hàng cách nhau bởi dấu cách
  + Dòng cuối cùng là một số nhị phân chỉ ra cách sắp xếp hàng đầu tiên. 0 là giảm dần, 1 là tăng dần.
* Đầu ra: ma trận đã sắp xếp, in ra trên m dòng, mỗi dòng gồm n phần tử cách nhau bởi dấu cách.
* Ràng buộc: các phần tử trong mảng là số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4  3 1 2 3  4 6 4 5  1 3 2 1  1 | 1 2 3 3  6 5 4 4  1 1 2 3 |
| 4 3  3 2 4  5 6 3  6 3 4  1 1 1  0 | 4 3 2  3 5 6  6 4 3  1 1 1 |

Giải:

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

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

sorting\_order\_first\_row = int(input())

matrix[0].sort(reverse=sorting\_order\_first\_row == 0)

for i in range(1, m):

if i % 2 == 0:

matrix[i].sort(reverse=not sorting\_order\_first\_row)

else:

matrix[i].sort(reverse=sorting\_order\_first\_row)

for row in matrix:

print(" ".join(map(str, row)))

Bài 17:

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tích 2 ma trận X.Y và in kết quả ra màn hình. Nếu không tính được tích 2 ma trận thì in ra thông báo "Du lieu vao sai"

Input:

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

Output:

+ Nếu tính được tích:

- Dòng thứ nhất in ra thông báo "Ma tran tich"

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tích:

- In ra thông báo "Du lieu vao sai"

Constranins: 1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 8  4 0 9 4 8 8 2 4  5 5 1 7 1 1 5 2  8 8  6 1 4 2 3 2 2 1  6 8 5 7 6 1 8 9  2 7 9 5 4 3 1 2  3 3 4 1 1 3 8 7  4 2 7 7 9 3 1 9  8 6 5 0 2 8 6 0  2 4 8 6 5 0 9 0  0 6 1 3 8 9 3 4 | Ma tran tich  154 175 229 137 182 171 135 138  105 113 136 100 108 68 165 118 |

Giải:

def multiply\_matrices(X, Y):

result = []

if len(X[0]) != len(Y):

print("Du lieu vao sai")

return

for i in range(len(X)):

row = []

for j in range(len(Y[0])):

element = 0

for k in range(len(Y)):

element += X[i][k] \* Y[k][ j ]

row.append(element)

result.append(row)

print("Ma tran tich")

for row in result:

print(\*row)

n1, m1 = map(int, input().split())

X = [list(map(int, input().split())) for \_ in range(n1)]

n2, m2 = map(int, input().split())

Y = [list(map(int, input().split())) for \_ in range(n2)]

multiply\_matrices(X, Y)

Bài 18:

Cho một dãy số nguyên. Hãy chia dãy thành hai phần: phần bên trái gồm các số âm sắp xếp theo thứ tự giảm dần, phần bên phải gồm các số không âm sắp xếp theo thứ tự tăng dần.

* Đầu vào: Dãy các số nguyên cách nhau bởi dấu cách
* Đầu ra: Dãy sau khi đã sắp xếp

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 3 2 -1 -4 0 | -1 -4 0 2 3 5 |

Giải:

def sap\_xep\_duong\_am(arr):

left\_part = sorted(filter(lambda x: x < 0, arr), reverse=True)

right\_part = sorted(filter(lambda x: x >= 0, arr))

result = left\_part + right\_part

return result

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

result = sap\_xep\_duong\_am(arr)

print(\*result)

Bài 19:

**Bạn hãy viết hàm: Kiểm tra các chữ số của một số nguyên dương n có toàn là số chẵn không.**

**Viết chương trình nhập vào ba số nguyên x, y, z. In ra "Yes" nếu các chữ số của x (hoặc y, z) toàn là số chẵn. In ra "No" nếu các chữ số của x (hoặc y, z) có ít nhất một số lẻ. In ra "Error" nếu x (hoặc y, z) là số âm. Thông báo trên cùng một dòng, cách nhau dấu cách.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output: No Error Yes**

**Example 2:**

**+ Input: 190 172 111**

**+ Output: No No No**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | No Error Yes |

Giải:

def kiem\_tra\_so\_chan(n):

if n < 0:

return "Error"

for digit in map(int, str(n)):

if digit % 2 != 0:

return "No"

return "Yes"

x,y,z=map(int,input().split())

result\_x = kiem\_tra\_so\_chan(x)

result\_y = kiem\_tra\_so\_chan(y)

result\_z = kiem\_tra\_so\_chan(z)

print(result\_x, result\_y, result\_z)

Bài 20:

**Bạn hãy viết hàm: Tìm chữ số lớn nhất của một số nguyên dương n.**

**Viết chương trình nhập vào ba số nguyên x, y, z. In ra các chữ số lớn nhất tìm được của x, y, z. In ra "Error" nếu x (hoặc y, z) là số âm. Thông báo trên cùng một dòng, cách nhau dấu cách.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output: 9 Error 8**

**Example 2:**

**+ Input: 190 172 115**

**+ Output: 9 7 5**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | 9 Error 8 |

Giải:

def tim\_chu\_so\_lon\_nhat(n):

if n < 0:

return "Error"

max\_digit = max(map(int, str(n)))

return str(max\_digit)

x,y,z=map(int,input().split())

result\_x = tim\_chu\_so\_lon\_nhat(x)

result\_y = tim\_chu\_so\_lon\_nhat(y)

result\_z = tim\_chu\_so\_lon\_nhat(z)

print(result\_x, result\_y, result\_z)

Viết chương trình nhập vào 2 số nguyên a, b. In ra các số nguyên tố trong đoạn [a, b] hoặc [b, a]. Nếu trong đoạn này không có số nguyên tố thì in ra thông báo "Khong co". Biết số nguyên tố là số nguyên dương >=2 và chỉ chia hết cho 1 và chính nó.

Input: a, b là số nguyên cách nhau dấu cách.

Output: các số nguyên tố tìm được viết trên một dòng, cách nhau dấu cách. Hoặc thông báo "Khong co".

Constrains: các số kiểu int

Ví dụ 1:

+ Input

a= -30, b=-9

+ Output

Khong co

Ví dụ 2:

+ Input

a= 0, b=9

+ Output

2 3 5 7

Ví dụ 2:

+ Input

a= 9, b=0

+ Output

7 5 3 2

**For example:**

| **Input** | **Result** |
| --- | --- |
| -9 -33 | Khong co |
| 28 18 | 23 19 |

import math

def snt(n):

if n <= 1:

return False

for i in range(2, int(math.sqrt(n)) + 1):

if n % i == 0:

return False

return True

a, b = map(int, input().split())

start, end = min(a, b), max(a, b)

lst = []

for i in range(start, end + 1):

if snt(i):

lst.append(i)

if not lst:

print("Khong co")

else:

if a > b:

lst.sort(reverse=True)

else:

lst.sort()

print(\*lst)

Viết hàm nhận đầu vào là số nguyên h khác 0. In ra hình vuông rỗng với cạnh bằng các ký tự \* theo quy tắc sau.

Ví dụ: h=5 thì hình vuông cần vẽ là

\*\*\*\*\*

\*    \*

\*    \*

\*    \*

\*\*\*\*\*

Viết chương trình nhập vào 3 số nguyên x, y, z. Dùng hàm trên để vẽ các hình vuông với chiều cao tương ứng với |x|, |y|, |z|.

Input: 3 số nguyên x, y, z cách nhau dấu cách.

Output:

Dòng 1 in số nguyên x

Từ dòng 2 in ra hình vuông với cạnh |x|

Tiếp theo đến y, z

Constrains: các dữ liệu là kiểu số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 -8 1 | 6  \*\*\*\*\*\*  \* \*  \* \*  \* \*  \* \*  \*\*\*\*\*\*  -8  \*\*\*\*\*\*\*\*  \* \*  \* \*  \* \*  \* \*  \* \*  \* \*  \*\*\*\*\*\*\*\*  1  \* |

def drawSquare(h):

h=abs(h)

if h==1:

print("\*")

return

if h<=0:

return 0

print("\*" \*h)

for \_ in range(h-2):

print("\*"+ " " \* (h-2)+ "\*")

print("\*" \*h)

x,y,z=map(int,input().split())

for v in x,y,z:

print(v)

drawSquare(v)