Exercice 1 : Nombres amicaux

a et b deux entiers positifs sont dits amicaux ou amiables ou aimables si chacun des deux nombres est égal à la somme des diviseurs stricts(diviseurs autres que lui-même) de l'autre .

Exemple :

- la somme des diviseurs de 220 (en excluant 220) : 1+2+4+5+10+11+20+22+44+55+110=284
- la somme des diviseurs de 284 (en excluant 284) : 1+2+4+71+142=220

Écrire un algorithme et une implémentation en python d'un programme nommé Amicaux qui permet de saisir deux entiers **m** et **n** et affiche s'ils sont amicaux ou non amicaux

Exemple d'exécution :

Veuillez entrer un entier n1 : 220

Veuillez entrer un entier n2 : 284

220 et 284 sont des nombres amicaux

Exercice 2 :

Ecrire l'**algorithme** ainsi que le **programme en Python** du problème qui permet de :

- ✓ Saisir une chaîne des caractères **ch** représentant un nombre complexe de la forme a+bi avec a et b sont deux entiers naturels (on suppose que **ch** est valide)
- ✓ Calculer et afficher le module du nombre complexe **ch**

N.B : le module d'un nombre complexe de la forme **a+bi** est égal à $\sqrt{a^2 + b^2}$

Exemples :

Donnée :	***Résultat :***
1) ch = ″4+3i″	module = 5
2) ch = ″2+15i″	module = 15.13

Execice 2:

En algorithmique	Implémentation en python

Algorithme du programme principal (PP)

Algorithme racinecarré
Début
ch← Saisir( )
m←module(ch)
afficher(ch,m)
Fin

TDO globaux

Objet	type
ch	**Chaine de caractère**
module, Saisir	**Fonction**
afficher	**Procédure**

Algorithme de la fonction module

Fonction module(ch**: Chaine) :entier**
Début
P← pos('+', ch)
a← **valeur**(sous chaine(ch,0**,**p))
b← **valeur**(sous chaine(ch, p+1**,** len(ch)-1))
m← racinecarré(a*a + b*b)

retourner m
fin

TDO locaux

Objet	type
a,b,p	**Entier**

Algorithme du Procédure afficher

Procédure affiche (ch :chaine ,m :entier)
Début
Ecrire(('le module de ',ch,'est',m)
Fin

```
from math import *
def  saisir( ):
    ch=input('donner chaine: ')
    return ch

def  module(ch):
    p= ch.find('+')
    a= int(ch[0:p])
    b= int(ch[p+1:len(ch)-1])
    m= sqrt(a*a + b*b)
    return (m)

def  afficher(ch,m):
    print('le module de ',ch,'est',m)

# Program principal
ch=saisir( )
m=module(ch)
afficher(ch,m)
```

Exercice 1 : Nombres amicaux

En algorithmique

Algorithme du programme principal (PP)

Algorithme amicaux
Début
n= Saisir()
m= Saisir()
Si nombres_amicaux(n,m) **alors**
Ecrire('nombres amicaux')
Sinon
Ecrire('nombres non amicaux')
Fin si
Fin

TDO globaux

Objet	type
n,m	**Entier**
nombres_amicaux	**fonction**

Algorithme de la fonction nombres_amicaux

Fonction nombres_amicaux(p,q :entier) :booleén
Début
a←somme_diviseurs(p)
b← somme_diviseurs(q)
si a=q et b=p a**lors**
retourner vrai
sinon
retourner faux
fin si
fin

TDO locaux

Objet	type
a,b	**Entier**
somme_diviseurs	**Fonction**

Fonction somme_diviseurs(p :entier) :entier
Début
somme←0
Pour i de 1 à p Div 2 **faire**
Si p mod i=0 **alors**
somme←somme+i
Finsi
Fin pour
Retourner somme
Fin

TDO locaux

Objet	type
i,somme	**entier**

Implémentation en python

```
def saisir():
    x=int(input('donner un entier '))
    while not(0<x):
        x=int(input('donner un entier '))
    return x

def Somme_diviseurs(p):
    somme=0
    for i in range(1,(p//2)+1):
        if p%i==0:
            somme=somme+i
    return somme


def nombres_amicaux(p,q):
    a=Somme_diviseurs(p)
    b=Somme_diviseurs(q)
    if a==q and b==p:
        return True
    else:
        return False

#Program principal

n=saisir()
m=saisir()
if nombres_amicaux(n,m):
        print('nombres amicaux')
else:
        print('nombres non amicaux')
```