1.- (15 puntos) Para su tarea de matemáticas, Euclides necesita que, dado dos puntos del espacio bidimensional, $a = (a_x, a_y)$ y $b = (b_x, b_y)$, se calcule el punto medio según la fórmula:

$$\frac{a+b}{2} = \left(\frac{a_x + b_x}{2}, \frac{a_y + b_y}{2}\right)$$

e identificar a qué cuadrante del plano pertenece dicho punto medio. Considere que un punto $p = (p_x, p_y)$ se encuentra en el primer cuadrante si $p_x > 0$ y $p_y > 0$, el punto p se encuentra en el segundo cuadrante si $p_x > 0$ y $p_y < 0$, el punto p se encuentra en el tercer cuadrante si $p_x < 0$ y $p_y < 0$, el punto p se encuentra en el cuarto cuadrante si $p_x < 0$ y $p_y > 0$, en otro caso, el punto p se encuentra en un eje.

Entrada: La entrada al programa consiste en 4 números enteros sin restricción de valor, correspondientes a las coordenadas a_x, a_y, b_x, b_y

Salida: La salida del programa es uno de los siguientes mensajes El punto medio es (p_x, p_y) - C cuadrante o El punto medio es (p_x,p_y) - Eje

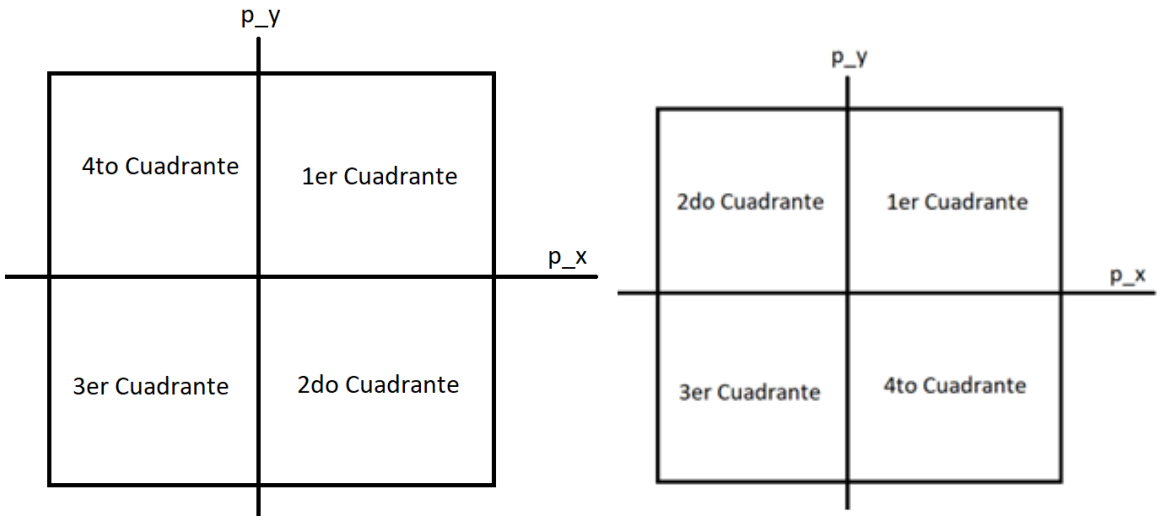
Ejemplo de Entrada: 5 -5 9-6

Ejemplo de Salida: El punto medio es (7.0, -5.5) - 4to cuadrante

(Creo que el enunciado estaba mal escrito)

Bonus: Se agregarán 5 puntos por usar una función que calcule el punto medio.

Datos: Observar bien, ¿Cuál Grafica hay que utilizar?



Funciones Utilizadas:

```
def validar(Min,Max,Txt):
    val=int(input(Txt))
    while val<Min and val>Max:
        val=(int(input("Error, Valor fuera de rango:")))
    return val
```

Función utilizada para confirmar que un dato está en los intervalos pedidos [Min, Max], y en el caso de que este valor no esté en el rango, se pedirá ingresar un nuevo valor.

```
#Punto medio en el Eje X
def p_x(x,y):
    a=0
    a=(x+y)/2
    return a
#----------------------------#
#Punto medio en el Eje Y
def p_y(x,y):
    a=0
    a=(x+y)/2
    return a
```

Funciones utilizadas para obtener los valores de los ejes del punto medio

```
def cuadrante(x,y):
    if x>0: #Positivo
        if y>0: #Positivo
            print("1er Cuadrante")
        elif y<0: #Negativo
            print("2do Cuadrante")
        else:
            print("Eje")
    elif x<0: #Negativo
        if y>0: #Positivo
            print("4to Cuadrante")
        elif y<0: #Negativo
            print("3er Cuadrante")
        else:
            print("Eje")
    else:
        print("Eje")
```

Función que nos permitirá saber en que cuadrante se ubica el punto medio

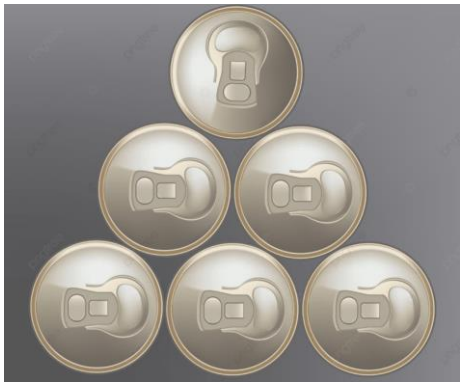
Ojo: Observar bien las condiciones de los cuadrantes.

Código Principal:

```
a_x=int(input("a_x="))
a_y=int(input("a_y="))
b_x=int(input("b_x="))
b_y=int(input("b_y="))
print("El punto medio es ",p_x(a_x,b_x),",",p_y(a_y,b_y) ,end=' - ')
print(cuadrante(p_x(a_x,b_x),p_y(a_y,b_y)))
```

Pedimos las componentes de los puntos ‘a’ y ‘b’, luego mediante el uso de las funciones calculamos el punto medio, y la posición del mismo (cuadrante).

2.- (20 puntos) En el supermercado, el reponedor coloca las latas de conserva apiladas triangularmente, esto es, en último piso una lata, en el penúltimo piso dos latas, en el antepenúltimo piso tres, y así sucesivamente. Por ejemplo, seis latas se apilan así:



Sin embargo, no todo número de latas se puede apilar triangularmente. Realice un programa en Python, en el que dada una cantidad de latas n, introducido por el usuario, compruebe si se pueden apilar.

Entrada: La única entrada al programa contiene un entero n (n ≥ 1) que representa la cantidad de latas. Se debe validar la entrada, esto significa que si su valor está fuera de rango se debe repetir sucesivamente hasta que el ingreso sea correcto.

Salida: La única salida puede ser uno de los siguientes mensajes Las n latas se pueden apilar en q filas o No apilables. Las n latas se pueden apilar en q filas pero sobran s latas.

Ejemplo de Entrada 1: 6

Ejemplo de Salida 1: Las 6 latas se pueden apilar en 3 filas

Ejemplo de Entrada 2: 8

Ejemplo de Salida 2: No apilables. Las 8 latas se pueden apilar en 3 filas, pero sobran 2 latas

Datos:

Observamos que a cada fila se le agrega una lata mas

Visualización del Ejemplo 2: n=8 latas

Funciones Utilizadas:

```
def validar(Min,Max,Txt):
    val=int(input(Txt))
    while val<Min and val>Max:
        val=(int(input("Error, Valor fuera de rango:")))
    return val
```

Función utilizada para confirmar que un dato está en los intervalos pedidos [Min, Max], y en el caso de que este valor no esté en el rango, se pedirá ingresar un nuevo valor.

```
def pisos_max(latas):
    pisos=0
    while latas>0:
        pisos=pisos+1
        latas=latas-pisos
    return pisos
```

Función que nos permite ver cuántos piso máximo va a tener la pirámide (considerando hay las latas suficientes para llenarlo)

```
def suma_latapiso(pisos_max):
    latas=0
    latas_max=0
    while pisos_max!=0:
        pisos_max=pisos_max-1
        latas=latas+1
        latas_max=latas_max+latas
    return latas_max
```

Función que nos permite saber cuantas latas hay en nuestra supuesta torre, con el valor de los pisos máximos calculados anteriormente.

```
def dif_latas(suma_latapiso,cant_latas,pisos_max):
    if suma_latapiso>cant_latas:
        print(f"No apilables. Las {cant_latas} latas se pueden apilar en {pisos_max-1}",end=" ")
        print(f"filas pero sobran {suma_latapiso-cant_latas} latas")
    else:
        print("Las",cant_latas,"latas se pueden apilar en",pisos_max,"filas")
```

Función que nos mandara el resultado final, de si es o no apilable, y de no ser apilable, nos dirá cuantas latas sobraron.

Código Principal:

```
n=validar(1,float("inf"),"cantidad de latas:")
dif_latas(suma_latapiso(pisos_max(n)),n,pisos_max(n))
```

Pedimos el numero de latas, y luego nuevamente hacemos uso de las funciones antes definidas, para obtener el resultado.

3.- (25 puntos) Organilada® es una fábrica de mermeladas orgánicas que, en vez de cosechar del árbol, fabrica mermeladas solo con las frutas que se caen por sí solas, procesándolas apenas tocan el suelo. El problema que tiene Organilada® es que no puede almacenar frascos de mermelada vacíos y desconoce la cantidad que podrá llenar en un día, por tanto, no puede mandarlos a pedir con anticipación. Para resolver este problema se le pide escribir un programa en Python que reciba información sobre las frutas recolectadas y calcule la cantidad de frascos necesarios para embotellar toda la mermelada.

Entrada: Las entradas del programa consiste en varias líneas. La primera contiene T, un entero positivo menor o igual a 1000, que representa la capacidad (en gramos) de los frascos disponibles. La segunda línea contiene M, un entero no negativo menor o igual a 200, el cual representa la cantidad de mermelada (en gramos) que se obtiene por cada fruta. Luego, el programa debe aceptar el ingreso de un conjunto de valores enteros positivos, que representan la cantidad de frutas que caen por día. Las entradas terminan cuando se encuentra un -1. Si cualquiera de las entradas está fuera de rango, el programa debe desplegar el mensaje "Error, valor fuera de rango", y dar la posibilidad de reingresar el valor.

Salida: El programa debe desplegar la cantidad máxima diaria de frascos que se pueden llenar de mermelada, considerando que puede haber quedado mermelada de los días anteriores. Además, al terminar el ingreso de frutas, el programa debe imprimir la cantidad restante de mermelada sin embotellar.

Ejemplo de Entrada:

750 (capacidad del frasco en gramos)

100 (cantidad de gramos de mermelada que produce una fruta)

5 (fruta caída el 1er día)

7 (fruta caída el 2do día)

10 (fruta caída el 3er día)

2 (fruta caída el 4to día)

-1 (fin del código)

Ejemplo de Salida:

0 (frascos llenos el 1er día, sobran 500 gramos de mermelada)

1 (frascos llenos el 2do día, sobran 450 gramos de mermelada)

1 (frascos llenos el 3er día, sobran 700 gramos de mermelada)

1 (frascos llenos el 4to día, sobran 150 gramos de mermelada)

150 (gramos de mermelada sobrante)

Funciones Utilizadas:

```
def validar(Min,Max,Txt):
    val=int(input(Txt))
    while val<Min and val>Max:
        val=(int(input("Error, Valor fuera de rango:")))
    return val
```

Función utilizada para confirmar que un dato está en los intervalos pedidos [Min, Max], y en el caso de que este valor no esté en el rango, se pedirá ingresar un nuevo valor.

Código Principal:

```
T=validar(1,1000,"Capacidad(en gramos) de los frascos:")
M=validar(0,200,"Cantidad de Mermelada(en gramos) por cada fruta:")
a=1
b=0
while a!=0: #ciclo Infinito
    c=0
    x=validar(-1,float("inf"),"Fruta caida ese dia:")
    if x!=-1:
        print(x,"Fruta(s)",",",x*M,"gramos de Mermelada Producida, Hoy")
        b=x*M+b
        while b>T:
            c=c+1
            b=b-T
        print(c,"Frascos llenos, Hoy")
    else: #x==-1
        break
print(b,"gramos Mermelada sobrante, sin enbotellar")
```

Seleccionamos con cuantos gramos llenamos un frasco de mermelada (T) y luego la cantidad de gramos de mermelada de produce una fruta (M; 1 fruta = M gramos de mermelada)

Luego vamos seleccionando las frutas que cayeron en un día, y las convertimos en mermelada, que luego almacenamos

Si tenemos la suficiente mermelada para llenar un frasco, entonces llenaremos el frasco y contaremos cuantos frascos llenamos y cuanta mermelada nos sobro, pero solo anunciaremos cuantos frascos se hicieron ese día.

Finalmente, si ingresamos “-1” en el código, este se detendrá y nos mostrara cuanta mermelada quedo sin utilizar.