Ayudantía Programación

Certamen 1

Gonzalo Fernández

IWI-131 26-09-2017

Temas de la Ayudantía de Hoy

- Presentación de la Ayudantía
- Diagramas de Flujo
- Programación en Python
- Patrones de Análisis y Resolución de Problemas
- ~~Top 10 Funciones más Importantes en Python~~
- ~~Patrones de Programación~~
- ~~Prepárense para los Problemas~~ ~~(y más vale que teman)~~

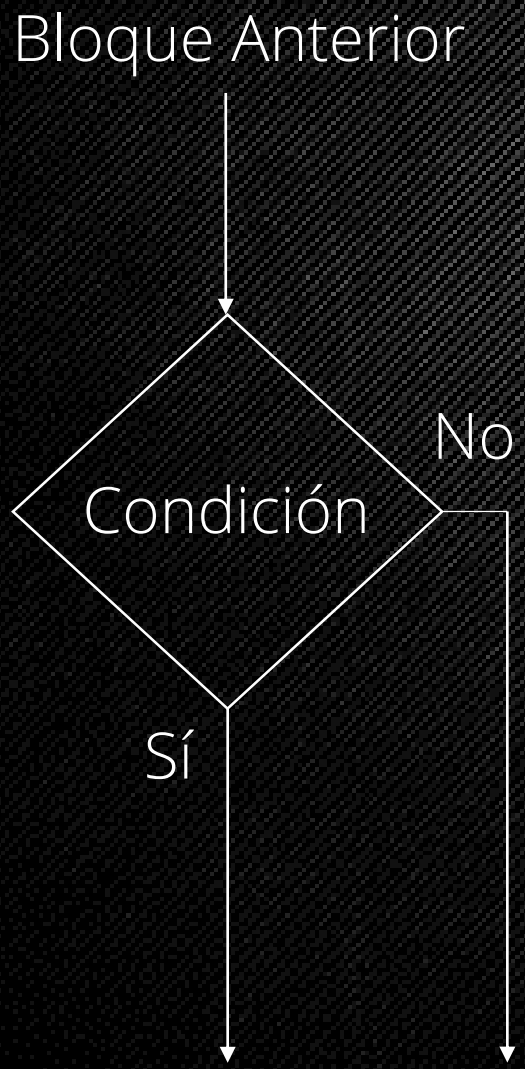
Presentación de la Ayudantía

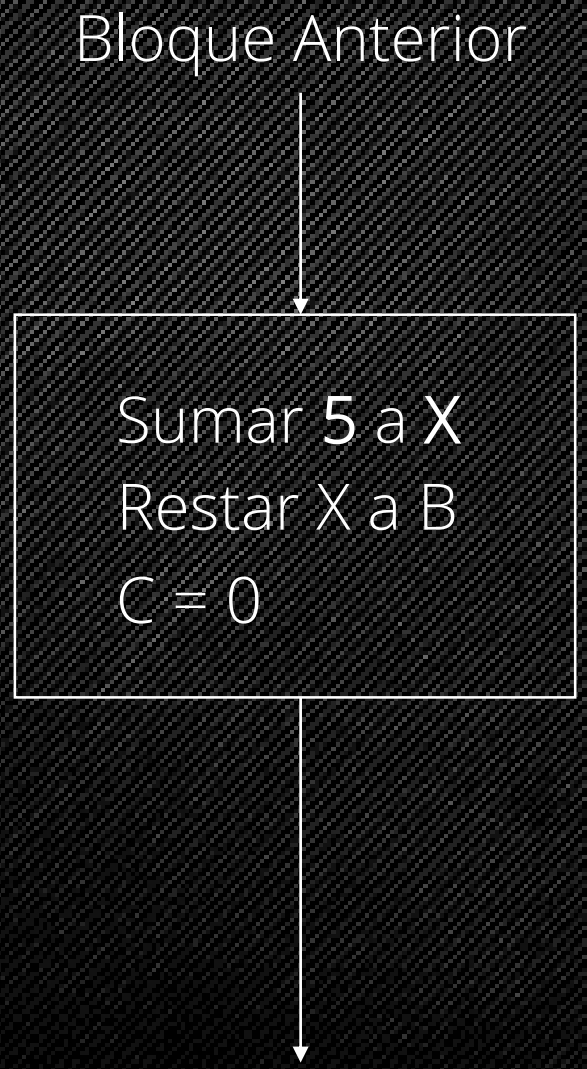
- **Ayudante**: Gonzalo Andrés Fernández Carrillo
- **Correo**: gonzalo.fernandezc@sansano.usm.cl
- **Horario**: Martes 9 – 10
- **Paralelo**: 213
- **Controles Lab**: 3
- **Repositorio GitHub**: https://git.io/vdId3

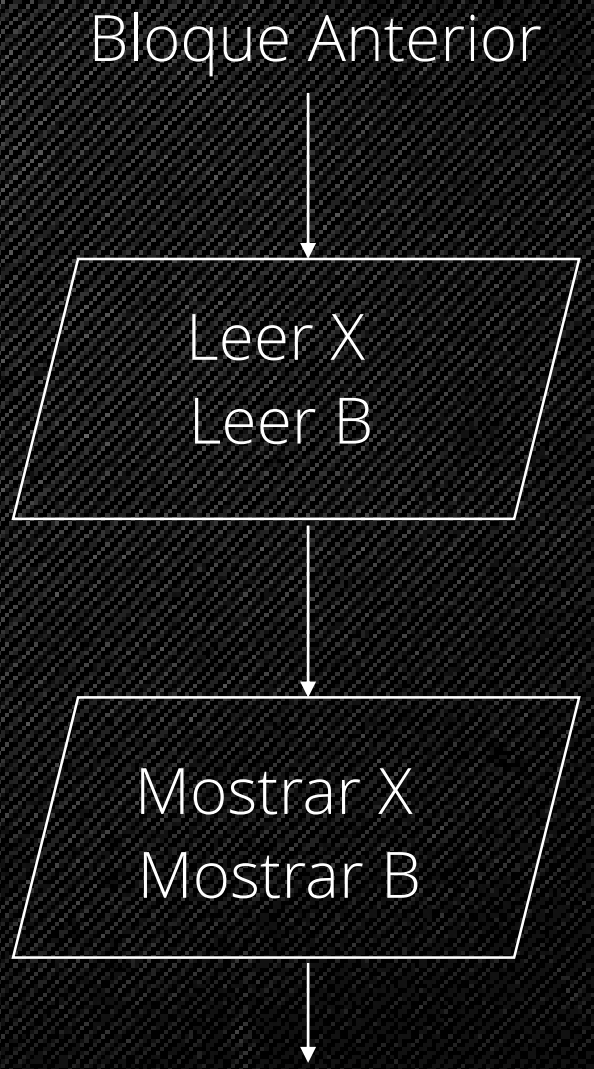
Diagramas de Flujo

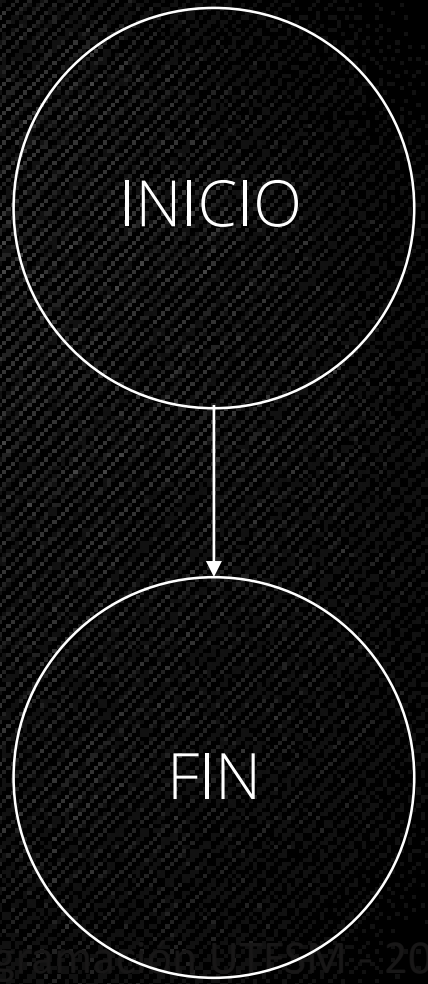
- Representación Gráfica de un **Algoritmo**
- Se compone de **4** bloques con distintas funciones:
  - Inicio – Fin
  - Condición
  - Acción
  - Lectura – Salida
- Cada unión entre bloques debe indicar su **Dirección** mediante una flecha
- Todo Diagrama debe indicar su **Inicio** y **Fin**

Diagramas de Flujo









Programación en Python

Leer Strings:

x = raw_input (“”)

Leer Números:

x = int (raw_input (“”))

Leer Flotantes

x = float (raw_input (“”))

Mostrar Texto en Pantalla con Variables:

print “La variable X tiene el valor”, x, “Y fue leída”
print “La variable X tiene el valor ”+x+“ Y fue leída”

Programación en Python

Funciones:

```
def funcion ( parametro1, parametro2 ) :
---- parametro1 = parametro2
----
----
---- return parametro1
```

Programación en Python

- **Condición “Sí”:**

If (condición) :

- **Condición “Sí, de lo contrario”:**

If (condición) :

----

else:

- **Condición “Sí, o sí…, ó…”:**

If (condición) :

----

elif:

----

else:

Programación en Python

Ciclo For:

```
for i in datos:
---- print i
```

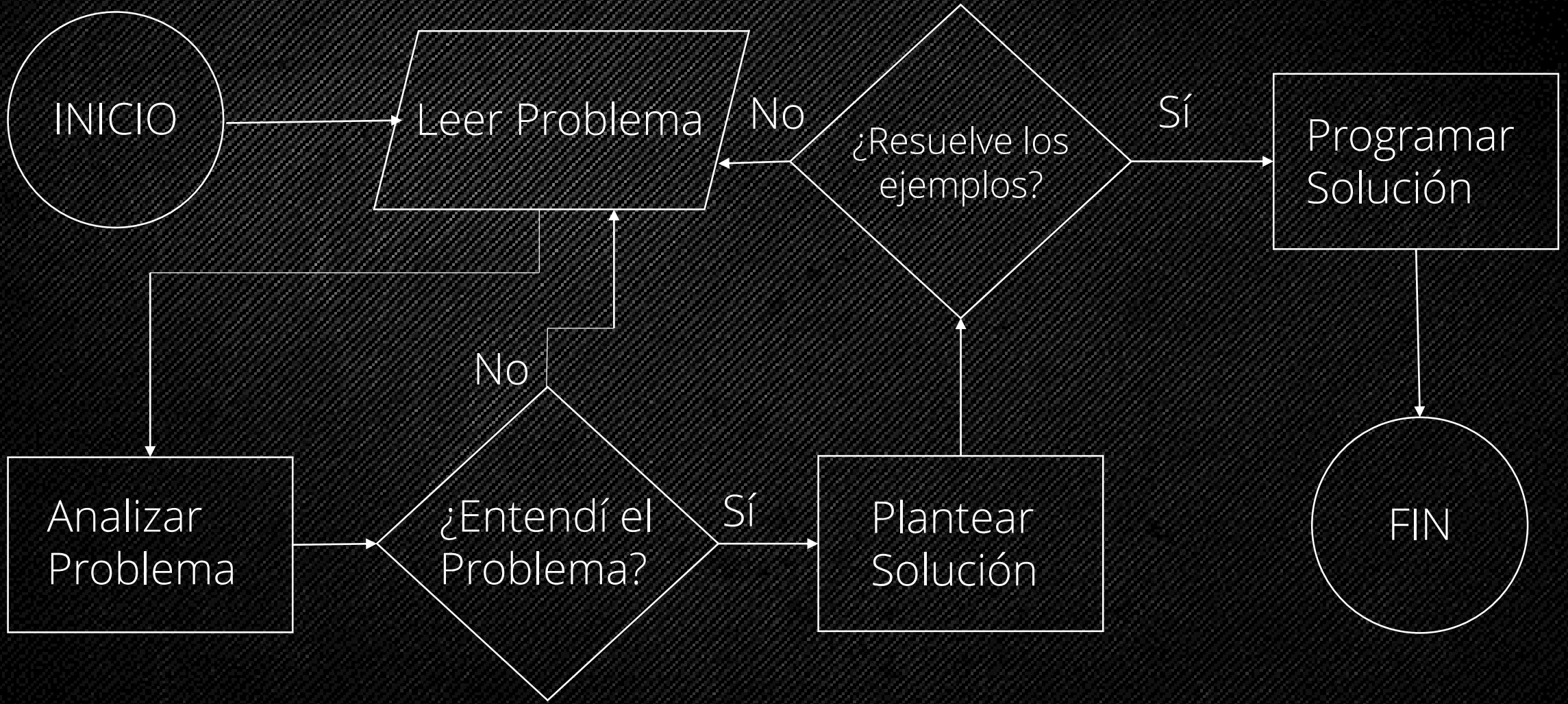
Ciclo While:

```
while ( condición ):
----
```

Patrones de Análisis y Resolución de Problemas

- Algoritmos base de problemas en general
- No resuelven el problema completo, pero sientan las bases para su análisis y desarrollo
- Básicamente, resuelven los sub-problemas del problema general
- Son fáciles de encontrar y de programar al momento de implementarlos

Súper Diagrama Resuelve Problemas



Patrones de Análisis de Problemas

Toda solución comienza con un correcto entendimiento del problema, hecho esto, se procede a pensar en casos “borde”, ó, que no estén a simple vista dentro del enunciado.

Estos casos pueden transformar una solución “obvia” en un tema un tanto más complicado, pues nos obliga a establecer más condiciones en nuestro algoritmo para evitar fallar en casos así.

Comprender el problema => Resolver el problema

Programación—Certamen 1 - Jueves 7 de Abril de 2016

Nombre: _____ Rol: _____ – _____

3. [40 %] Un grupo de bio-tecnólogos externos a la USM, tienen demasiado trabajo analizando cadenas de ADN. Todo el trabajo lo realizan a mano ya que no tuvieron un buen ramo de programación. Por esto, le piden a los estudiantes de el ramo de Programación IWI-131 que les ayuden. Las cadenas de ADN están compuestas por 4 bases nitrogenadas (`A`: Adenina, `C`: Citosina, `G`: Guanina y `T`: Timina) agrupadas en bloques de 4. Ahora usted debe ayudar en las siguientes operaciones:

a) Escriba la función `valida(cadena)` que reciba un string con una `cadena` de ADN y retorne **`True`** si la cadena es válida o **`False`** si no lo es. Una cadena no es válida cuando aparecen bases nitrogenadas distintas a las antes descritas.

```
>>> valida('CTGA CTGA AATT GGGC CTGG CCCC')
True
>>> valida('CTGA XCGA CGAT GGTA ACCC CCPC TTAA')
False
```

Solución “Obvia”

```
def validar (cadena):
    for i in cadena:
        if i != "C" and i != "T" and i != "G" and i != "A":
            return False
    return True
```

¿ Cual es el problema en esta solución ?

Patrones de Análisis de Problemas

Ahora bien, un problema normalmente no es fácil de entender en la primera lectura, por lo que a medida que leemos el enunciado, es necesario discernir de lo que no nos aporta datos y centrarnos en todos los rasgos que aportan al problema.

Normalmente estos rasgos son:

- Datos de Entrada
- Valores
- Condiciones Explícitas
- Datos de Salida

Programación—Certamen 1 - Miércoles 12 de Abril de 2017

Nombre

Rol — Paralelo

2. [35 %] La constructura Pythonia tiene en su haber una gran cantidad de terrenos y su negocio es venderlos por lotes para que empresas y particulares construyan sus fábricas o viviendas. La manera actual de establecer los precios de esos lotes de terreno es preguntándole a un tasador. Debido a la gran cantidad de terrenos que la constructora está teniendo, se requiere un algoritmo para que establezca los precios puesto que el tasador ya no da abasto.

El precio por metro cuadrado es de 20 UF. Si el área del terreno es inferior a 100m2 se aplica un descuento de 10 %, si el área esta comprendida entre 100m2 y 1000m2, se le aplica un incremento del 20 %. Finalmente, si es superior a 1000m2 se suma un 50 %. Además los terrenos están en 3 sectores distintos.

Haga un diagrama de flujo que vaya preguntando por el área y sector de cada terreno a tasar. Cuando el área ingresada sea cero, el programa debe finalizar, pero antes debe imprimir por cada sector el total avaluado junto con el área y precio del terreno más caro en éste.

Patrones de Análisis de Problemas

Datos de Entrada:

- Área de cada Terreno
- Sector de cada Terreno

Valores:

- Precio m^2: 20UF
- 3 Sectores distintos

Datos de Salida:

- Total avaluado de cada Sector
- Terreno más caro de cada Sector

Condiciones Explícitas:

- Área < 100m^2 => Descuento del 10%
- 100 m^2 ≤ Área ≤ 1000m^2 => Incremento del 20%
- Área > 1000m^2 => Suma del 50%
- Área == 0 => Imprimir Datos de Salida y Terminar Programa

Patrones de Análisis de Problemas

Ahora que comprendemos el problema a cabalidad, podemos proseguir a dividirlo en sub-problemas.

Un sub-problema nos permite abstraernos del problema en particular y pensar en una solución más general, que técnicamente sería una “sub-solución”, la correcta integración de todas las “sub-soluciones” nos permitirán resolver el problema completo.

No siempre es necesario realizar esto, pero suele ayudar mucho.

SubProblemas

1. Leer correctamente la Entrada de Datos
2. Almacenar correctamente la información que trabajaré
3. Identificar qué otros datos necesitaré almacenar
4. Implementar las condiciones extraídas del problema
5. Realizar un flujo correcto entre los datos y las condiciones
6. Mostrar correctamente la Salida de Datos

¿ Qué otro sub-problema crees que hay?

Patrones de Resolución de Problemas

Ya tenemos los problemas generales que vamos a resolver, así como una noción profunda del problema en particular, ahora debemos preguntarnos:

¿Cómo @#$/)$!# resuelvo este problema?

¿A qué otro problema que conozco se parece?

¿Existe alguna estructura general?

~~¿Qué es un problema?~~

~~¿Qué somos?~~

Patrones de Resolución de Problemas

En este problema existen varios sub-problemas conocidos, y que son, a la vez, fáciles de resolver:

1. Condiciones de intervalos
2. Procesar valores de una entrada de datos constante
3. Mantener el mayor valor en una entrada de datos constante

¿Qué otro sub-problema conocido pueden identificar?

Patrones de Resolución de Problemas

Si bien, no siempre es posible reducir todos los problemas en sub-problemas, y estos, a soluciones generales ya conocidas, con lo que hemos logrado ya podemos pasar a resolver el problema en sí, pues ya tenemos una base importante de la solución y los demás detalles de esta es necesario analizarlos caso a caso.

Actividad

Para el problema analizado anteriormente, realizar:

1. Solución en Diagrama de Flujo
2. Solución en Python
3. Ruteo de la Solución en Python

Actividad

1. Buscar problemas para resolver en la ayudantía “Extra”, de cualquier tipo, ruteos, diagramas de flujo, códigos, existenciales, etc.

2. Recopilar dudas de materia que puedan tener.
3. Recopilar cualquier cosa que necesiten en la que pueda ayudarlos para el Certamen 1

Enviar a mi correo: gonzalo.fernandezc@sansano.usm.cl

Ayudantía Recuperativa

- Resolución de Dudas y Problemas recopilados por Ustedes
- Top 10 Funciones más Importantes en Python
- Patrones de Programación
- Prepárense para los Problemas (y más vale que teman)

Ayudantía Recuperativa

- Resolución de Dudas y Problemas recopilados por Ustedes
- Top 10 Funciones más Importantes en Python
- Patrones de Programación
- Prepárense para los Problemas (y más vale que teman)

b) Escriba la función `cantidad(cadena, base)` que reciba un string con una `cadena` de ADN y una `base` nitrogenada. La función debe retornar la cantidad de apariciones de la `base` en la `cadena`.

Nota: no puede utilizar el método `count` de procesamiento de texto.

```
>>> cantidad('CTGA CTGA AATT GGGC CTGG CCCC', 'A')
4
```

c) Los científicos encontraron un patrón de clasificación, que se deduce de la cantidad mayoritaria de un cierto par de bases. Si la suma de las cantidades de Citosina y Guanina es mayor a la de Adenina y Timina, es una especie vegetal, en caso contrario es una especie animal. Escriba un programa que pregunte la cantidad de cadenas de ADN a evaluar, luego solicite las cadenas y finalmente muestre las cantidades de cada especie y cadenas no válidas.

```
Cantidad de cadenas de ADN: 3
Ingrese cadena 1: CGTA CAGT TTGG GGTA AATG CATG
Ingrese cadena 2: CACC CTGA GGAA ACAA XTFC ATGG
Ingrese cadena 3: TGTG TTGA ATGA CTAT ATTT
Cantidad animales: 2
Cantidad vegetales: 0
Cantidad no validas: 1
```

Nombre

Rol — Paralelo

3. [40 %] La comisión organizadora de la semana mechona creó un concurso on line, en el cual sus 3 alianzas deben participar. El concurso consta de una serie de turnos, en los cuales cada alianza debe ingresar una palabra. La palabra con mayor cantidad de vocales gana. Si dos o más alianzas igualan en el máximo de vocales, nadie gana esa ronda. El concurso termina cuando **una** alianza logre ganar x juegos (meta), la cual se debe definir al inicio del juego.

En base a lo anterior, la comisión le solicita a usted implementar lo siguiente:

a) Escriba la función `ganador(c1,c2,c3,meta)` la cual recibe 4 parámetros, la cantidad de juegos ganados por la alianza 1, 2 y 3 y la meta a lograr en el juego. La función debe retornar el número de la alianza ganadora. En caso de no existir ganador, debe retornar el valor entero 0 (cero).

Nota: Tener en cuenta que no pueden existir empates en las alianzas al momento de alcanzar la meta.

```
>>> ganador(2,5,3,5)
2
>>> ganador(1,3,3,5)
0
```

b) Escriba la función `contar(palabra)` que reciba un string `palabra`. La función debe retornar la cantidad de vocales existentes en la `palabra` recibida.

Nota: no puede utilizar el método `count` de procesamiento de texto.

```
>>> contar('paralelepipedo')
7
>>> contar('str')
0
```

c) Desarrolle un programa que solicite la meta del Juego, y luego solicite las palabras de cada alianza por turno, hasta que exista una alianza que logre la meta. **Asuma que las palabras siempre serán ingresadas en minúsculas**. A continuación se presenta un ejemplo de cómo debiera lucir el programa.

```
Ingrese meta del Juego: 3
alianza 1: paralelepipedo
alianza 2: reuna
alianza 3: salida
alianza 1: tartamudo
alianza 2: mauricio
alianza 3: semana
alianza 1: murcielago
alianza 2: rescate
alianza 3: salvavida
alianza 1: temperatura
alianza 2: sala
alianza 3: certamen
La alianza ganadora es: 1
```