PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

IIC1103 2015-2
Interrogación 1

Sábado 12 de septiembre de 2015
Tiempo: 2 horas 30 minutos

Indicaciones

La interrogación consta de cuatro preguntas. Sólo se reciben consultas sobre el enunciado durante los primeros 40 minutos. La última página contiene un recordatorio; no tienes permitido usar material de apoyo adicional. A menos que el enunciado indique lo contrario, en tus respuestas no necesitas validar que el usuario haya ingresado los datos de manera correcta. Las cuatro preguntas están pensadas para ser resueltas en forma sencilla con condicionales y ciclos. No es necesario utilizar otros mecanismos más avanzados.

Problema 1

Crea un programa que permita dibujar un rombo a partir de su ancho, el cual es ingresado por el usuario. Por simplicidad, debes dibujar un rombo solamente si el número ingresado es impar, e indicar al usuario que no puede dibujar el rombo si el número es par.

Nota: Se puede imprimir varias veces un texto multiplicándolo por un número. Por ejemplo, `print("o"*4)` imprime "`oooo`" y `print("hola"*2)` imprime "`holahola`".

Ejemplos de ejecución (independientes entre sí):

```
>>>
Este programa dibuja un rombo
Ingrese el ancho del rombo (debe ser impar): 7
   *
  ***
 *****
*******
 *****
  ***
   *

>>>
Este programa dibuja un rombo
Ingrese el ancho del rombo (debe ser impar): 6
Ingreso un numero par; no se como dibujarlo!
```

Problema 2

Un corredor se encuentra corriendo la Maratón de Santiago 2016. La maratón tiene `42.195` km (con punto decimal), y en ciertos puntos de la carrera hay carteles indicando los kilómetros recorridos junto a un reloj que indica las horas y minutos transcurridos desde el inicio de la carrera. Nuestro corredor desea terminar la carrera en 4 horas o menos, y cada vez que ve un cartel quiere saber si va bien o si debe correr más rápido.

Escribe un programa para ayudar a nuestro amigo. Tu programa debe recibir un número real positivo menor o igual que `42.195` que indica la distancia recorrida en kilómetros; luego debe recibir dos enteros positivos indicando respectivamente las horas y minutos transcurridos. Con esta información tu programa debe imprimir uno de los siguientes mensajes en pantalla.

- "Ya pasaron las 4 horas. Lo siento."
- "Faltan K kilometros para terminar. Corriendo asi llegaras en H horas y M minutos. Sigue asi."
- "Faltan K kilometros para terminar. Debes ir mas rapido y subir tu velocidad a V km/h para llegar en 4 horas o menos"

Los mensajes cumplen que K y V son números reales positivos, H es un entero mayor o igual a `0`, y M es un entero entre `0` y `59`, inclusive. No es necesario que tu programa valide que la entrada dada por el usuario es legal. La fórmula para calcular la velocidad es V = distancia/tiempo. Dada una velocidad V y una distancia D, el tiempo para recorrer D a esa velocidad se calcula como D/V. Abajo hay tres ejemplos de ejecución (son independientes; no necesitas poner una iteración en tu programa):

```
>>>
** MARATON DE SANTIAGO **
Distancia recorrida? : 40.32
Hora?: 5
Minutos?: 23
Ya pasaron 4 horas. Lo siento.
```

```
>>>
** MARATON DE SANTIAGO **
Distancia recorrida? : 21.0975
Hora?: 2
Minutos?: 0
Faltan 21.0975 kilometros para terminar. Corriendo asi llegaras en 2 horas y 0
minutos. Sigue asi.
```

```
>>>
** MARATON DE SANTIAGO **
Distancia recorrida? : 22
Hora?: 2
Minutos?: 50
Faltan 20.195 kilometros para terminar. Debes ir mas rapido y subir tu
velocidad a 17.31 km/h para llegar en 4 horas.
```

Problema 3

Escribe un programa en Python que reciba como datos de entrada dos fracciones, calcule la suma, la reduzca a su mínima expresión y muestre el resultado en pantalla. El ingreso de cada fracción se realiza con dos números enteros para representar el numerador y denominador respectivamente.

La tabla de abajo muestra algunas entradas y sus respectivas salidas. Observa que se usa el formato "numerador"/"denominador", excepto cuando la fracción representa al número `1`, en cuyo caso se escribe "`1`" y no "`1/1`". Puedes suponer que cada fracción entregada por el usuario es menor que `1`, por lo que el resultado de la suma siempre será menor que dos.

Fracción 1	Fracción 2	Resultado
3/8	1/2	7/8
1/15	2/3	11/15
2/6	1/6	1/2
2/7	5/7	1
7/8	2/3	37/24

Abajo hay tres ejemplos de ejecución independientes:

```
>>>
** SUMA DE FRACCIONES **
Diga el numerador 1: 3
Diga el denominador 1: 8
Diga el numerador 2: 1
Diga el denominador 2: 2
La suma es: 7/8

>>>
** SUMA DE FRACCIONES **
Diga el numerador 1: 2
Diga el denominador 1: 7
Diga el numerador 2: 5
Diga el denominador 2: 7
La suma es: 1

>>>
** SUMA DE FRACCIONES **
Diga el numerador 1: 1
Diga el denominador 1: 2
Diga el numerador 2: 3
Diga el denominador 2: 4
La suma es: 5/4
```

Problema 4

Escribe un programa que permita calcular la distancia de Hamming entre dos números enteros que tienen igual número de dígitos. Esta distancia se define como la suma de la diferencia en valor absoluto de cada uno de los dígitos de los dos números. La suma de las diferencias de los dígitos se puede calcular de derecha a izquierda o de izquierda a derecha. En caso que los números tengan distinta cantidad de dígitos, indique que no es posible calcular la distancia de Hamming.

Por ejemplo, la distancia entre los números 8239 y 4561, se calcula de la siguiente manera:

Distancia entre 8 y 4: 4
Distancia entre 2 y 5: 3
Distancia entre 3 y 6: 3
Distancia entre 9 y 1: 8

Con lo cual, la distancia de Hamming sería: 4 + 3 + 3 + 8 = 18.

Ejemplos de ejecución (independientes entre sí):

```
>>>
Este programa calcula la distancia de Hamming entre dos numeros
Ingrese numero 1: 8239
Ingrese numero 2: 4561
La distancia de Hamming es: 18

>>>
Este programa calcula la distancia de Hamming entre dos numeros
Ingrese numero 1: 93
Ingrese numero 2: 4561
No se puede calcular la distancia de Hamming
```

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

Recordatorio contenidos I1 - Segundo semestre 2015

1. Tipos de datos y operadores

Tipo de dato	Clase	Ejemplo
Números enteros	`int`	`2`
Números reales	`float`	`2.5`
Números complejos	`complex`	`2 + 3j`
Valores booleanos	`bool`	`True`/`False`
Cadenas de texto	`str`	`"hola"`

Operación	Descripción	Ejemplo
`+`	Suma	`2.3+5.4`
`-`	Resta	`45.45-10.02`
`-`	Negación	`-5.4`
`*`	Multiplicación	`(2.3+4.2j)*3`
`**`	Potenciación	`2**8`
`/`	División	`100/99`
`//`	División entera	`100//99`
`%`	Módulo	`10%3`

Prioridad (de mayor a menor): `()`; `**`; `*`, `/`, `//` o `%`; `+` o `-`.

Operación	Descripción	Ejemplo
`==` `(!=)`	Igual (distinto) a	`2==2`
`<` `(<=)`	Menor (o igual)	`1<1.1`
`>` `(>=)`	Mayor (o igual)	`3>=1`
`and`	Ambos `True`	`2>1 and 2<3`
`or`	Algún `True`	`2!=2 or 2==2`
`not`	Negación	`not True`

Prioridad (de mayor a menor): `()`; `or`; `and`; `not`; comparadores.

2. Funciones `int, float, str`

- `int(arg)` convierte `arg` a entero.
- `float(arg)` convierte `arg` a número real.
- `str(arg)` convierte `arg` a cadena de texto.

3. Función `print`

- Un argumento:
  `print(arg)`
- Dos o más argumentos:
  `print(arg1, arg2,arg3)`
- Eliminar salto de línea:
  `print(arg, end='')`

4. Función `input`

- `ret = input(texto)` guarda en `ret` un `str` ingresado.
- `ret = int(input(texto))` guarda en `ret` un `int` ingresado.
- `ret = float(input(texto))` guarda en `ret` un `float` ingresado.

5. `if/elif/else`

```
if <cond 1> :
    <codigo si se cumple cond 1>
    if <cond 1.1> :
        <codigo si se cumple 1.1>
    else :
        <codigo si no se cumple 1.1>
elif <cond 2> :
    <codigo si se cumple cond 2 pero no cond 1>
else :
    <codigo si no se cumple cond 1 ni 2>
```

6. `while`

```
while <condicion> :
    <codigo que se ejecuta mientras
      se cumple condicion>
```

7. `random.randint(i,j)`

Requiere importar módulo: `import random`.
Retorna un número entero entre `i` y `j`, ambos inclusive.