Pontificia Universidad Católica de Chile
Escuela de ingeniería
Departamento de Ciencia de la Computación
Primer Semestre del 2014

IIC1103 Introducción a la Programación

Midterm – solución

Instrucciones

- Lea atentamente las preguntas.
- Responda cada pregunta en hojas separadas.
- Si tiene alguna duda levante la mano y un ayudante o profesor lo ayudarán
- Está estrictamente prohibido el uso de material complementario como apuntes, libros, notas, etc.
- No puede usar ningún dispositivo electrónico como celulares, tablets, ipads, laptops, etc.

Pregunta 1

Escribe una función llamada ***trioPitagorico*** que reciba como parámetro un número entero positivo *a* y determine si existen otros dos números enteros positivos, *b* y *c*, tales que:

$$a > b > c$$

es decir, *b* y *c* son menores que *a* y además son distintos entre ellos, y

$$a^2 = b^2 + c^2$$

es decir, *a*, *b* y *c* forman un *trío pitagórico*: si *a* fuera la longitud de la hipotenusa de un triángulo rectángulo, entonces *b* y *c* serían las longitudes de los catetos correspondientes

La función debe devolver una lista con los dos valores, *b* y *c*, o bien una lista vacía, si *b* y *c* no existen para el *a* dado.

Ejemplo:

***trioPitagorico*(5)** debe devolver la lista **`[4, 3]`**, ya que $5^2 = 4^2 + 3^2$; pero ***trioPitagorico*(6)** debe devolver la lista vacía, ya que no hay dos números enteros positivos menores que 6 tales que la suma de sus cuadrados sea igual a $6^2 = 36$.

Solución

```
def trioPitagorico(a):
    encontrado = False
    b = 2
    while not encontrado and b < a:
        c = 1
```

```
        while not encontrado and c < b:
            if a*a == b*b + c*c:
                encontrado = True
            else:
                c = c+1
        if not encontrado:
            b = b+1
    if encontrado:
        return [b,c]
    else:
        return []

n = int(input("ingrese n :"))
print(trioPitagorico(n))
```

Pregunta 2

En ocasiones es útil contar el número de palabras y caracteres de un texto. Es por este motivo que se te pide:

a) Crear una función ***creaDiccionario*** que reciba un texto. La función debe retornar un diccionario en el cuál las palabras son la clave y el valor es la frecuencia (cantidad de veces que la palabra aparece en el texto).
Ejemplo:
Si el texto fuese "El evento de Microsoft research realizado en Viña del Mar ha sido todo un éxito. El evento ha sido en el Hotel Sheraton y han concurrido importantes investigadores de diferentes universidades. El próximo año volveremos a participar"

El diccionario que se retorna tendrá la siguiente forma:

```
D={'realizado': 1, 'Microsoft': 1, 'concurrido': 1, 'Viña': 1, 'un': 1,
'evento': 2, 'research': 1, 'Hotel': 1, 'a': 1, 'volveremos': 1, 'del':
1, 'éxito.': 1, 'próximo': 1, 'diferentes': 1, 'El': 3, 'todo': 1,
'año': 1, 'y': 1, 'importantes': 1, 'Mar': 1, 'ha': 2, 'participar': 1,
'de': 2, 'Sheraton': 1, 'han': 1, 'sido': 2, 'investigadores': 1, 'en':
2, 'universidades.': 1, 'el': 1}
```

Solución:

```
#Función auxiliar para limpiar una palabra
def limpia(palabra_sucia):
    p_limpia = ""
    for caracter in palabra_sucia:
        if caracter != "." and caracter != "," and caracter != ";" and
                                                    caracter != ":":
            p_limpia += caracter
    return p_limpia

#Pregunta A
```

```
def creaDiccionario (texto):
    d= {}
    for palabra in texto.split():
        palabra_limpia = limpia(palabra)
        if palabra_limpia not in d:
            d[palabra_limpia] = 1
        else:
            d[palabra_limpia] += 1
        return d
```

b) Crear una función ***analisisEstadistico*** que reciba un diccionario y retorne una tupla con la palabra más repetida y la frecuencia de esa palabra.

Ejemplo

Si la función recibe el diccionario D de la pregunta anterior entonces la función debe retornar ('El',3).

Solución

```
#Pregunta B
def analisisEstadistico(D):
    maximo = 0
    clave = ""
    for palabra in D:
        if D[palabra] > maximo:
            maximo = D[palabra]
            clave = palabra
        return (clave,D[clave])
```

c) Crear el programa principal que pida un texto al usuario y que invoque a las funciones ***creaDiccionario*** y ***analisisEstadistico.*** El mensaje que se debe desplegar debe ser "La palabra X tiene la mayor frecuencia con Y repeticiones en el texto", donde X es la palabra más repetida e Y la frecuencia.

Solución

```
#Pregunta C
t=input().lower()
diccionario = creaDiccionario (t)
tupla = analisisEstadistico(diccionario)
   print("La palabra ",tupla[0]," tiene la mayor frecuencia con
",tupla[1]," repeticiones en el texto")
```

Es importante que elimines los signos de puntuación (puntos, comas, dos puntos y punto y coma). Además, en el caso de que existan 2 o más palabras con la misma cantidad de repeticiones, elige cualquiera de ellas.

Por último, debes convertir cada palabra del texto a minúsculas.

Pregunta 3

Twitter es una red social que permite enviar mensajes de hasta 140 caracteres de largo. Para facilitar la clasificación de los mensajes es posible incluir hashtags, que son palabras antecedidas por un #. Con ellos el sistema genera una lista de Trending Topics, que son los hashtags más usados.

Se te pide programar un mini Twitter, con las siguientes funcionalidades:

contieneHashtag(mensaje): Recibe un mensaje y retorna **True** si contiene uno o más hashtags y **False** en otro caso. Por ejemplo, ***contiene_hashtag("Me encanta #programar")*** retorna True, y ***contiene_hashtag("Me encanta programar")*** debe retornar **False**.

obtenerHashtags(mensaje): Recibe un mensaje, y retorna todos los hashtags que tiene ese mensaje como un string separado con un espacio entre cada uno. Ejemplo: si recibe "Me encanta el #verano y #programar" debe retornar *verano programar*.

ponerHashtag(mensaje, hashtags): Recibe un string que representa el mensaje a publicar y un string compuesto por hashtags separados por comas (','). Debe retornar el mensaje con # insertados antes de las palabras recibidas en hashtags. Ejemplo, dado el mensaje "Me encanta el verano y programar" y *verano,programar*, debe retornar "Me encanta el #verano y #programar"

actualizarTrendingTopics(hashtag): Nuestro Twitter está interesado en contar solo los hashtags "verano", "programar" e "IIC1103". Esta función recibe una palabra y, en caso de ser alguna de las palabras de interés, actualiza la cantidad de menciones de ese hashtag. Es decir, si se llama a actualizar_trending_topics("programar"), debe aumentar en 1 la cantidad de menciones de la palabra "programar". Declara las variables adecuadas para llevar esta cuenta.

twittear(mensaje, truncar): Si *mensaje* contiene las palabras "verano", "programar" y/o "IIC1103" (sin hashtags), debes usar las funciones definidas anteriormente para agregar los hashtags al mensaje. Una vez hecho esto, o bien si el mensaje original ya tenía los hashtags, debes actualizar las menciones de cada trending topic y mostrar un mensaje con la cantidad de veces que se ha mencionado cada una de esas palabras. Si el mensaje no contiene las palabras de interés, no se debe mostrar el conteo de menciones.

Si el mensaje procesado como se señala anteriormente excede los 140 caracteres:

- Si truncar es **True**, retorna el mensaje hasta el caracter 140.
- Si truncar es **False**, retorna los caracteres sobrantes, desde el 141 en adelante

En caso de que el mensaje no exceda los 140 caracteres, se retorna el mensaje completo.

Puedes asumir que cada palabra de interés aparecerá a lo más una sola vez por mensaje. Recuerda, además, que las palabras de interés pueden venir en mayúscula o minúscula

Solución

```
lista_topicos = ['verano', 'programar', 'IIC1103']
conteos_verano = 0
conteos_programar = 0
conteos_iic1103 = 0


def twittear(mensaje, truncar):
    # Vemos si contiene hashtags
    #if not contiene_hashtag(mensaje):
    mensaje_hts = poner_hashtag(mensaje, 'verano,programar,IIC1103')
    # En este punto, el mensaje tendra # antecediendo
    # las palabras de interes
    topicos_mensaje = obtener_hashtags(mensaje_hts).split(' ')
```

```
    for topico in topicos_mensaje:
        print("Topico: ", topico)
        if topico in lista_topicos:
            actualizar_trending_topics(topico)

    #140 caracteres es la restriccion de Twitter
    if len(mensaje_hts) > 140:
        # Si el usuario quiere mandar el mensaje truncado
        if truncar:
            return mensaje_hts[:140]
        # Si el usuario quiere saber por cuanto se paso
        else:
            return mensaje_hts[140:]
    else:
        return mensaje_hts

def contiene_hashtag(mensaje):
    if mensaje.find("#") != -1:
        return True
    else:
        return False

def obtener_hashtags(mensaje):
    hashtags = ""
    i = 0
    if contiene_hashtag(mensaje):
        while i < len(mensaje):
            # Encontramos hashtag
            if mensaje[i] == "#":
                pos_espacio = mensaje[i:].find(" ")
                # Si es la ultima palabra
                if  pos_espacio == -1:
                    hashtags += mensaje[i+1:]
                else:
                    hashtags += mensaje[i+1 :i + pos_espacio] + " "
            i += 1
    # Sacamos los espacios sobrantes y retornamos
    return hashtags.strip()


# Recibimos los hashtags separados por ','
def poner_hashtag(mensaje, hashtags):
    lista_hashtags = hashtags.split(',')
    for i in lista_hashtags:
        # Pasamos a lowercase para poder
        # detectar correctamente las palabras
        i = i.lower()
        mensaje = mensaje.lower()
        # Si tiene palabras de interes que no
        # esten marcadas con #
        if mensaje.find(i) != -1 and mensaje[mensaje.find(i)-1] != '#':
            mensaje_separado = mensaje.split(i)
            # Insertamos el simbolo de hashtag, sacando los espacios que
            # pudiera tener la palabra en la frase original
            mensaje = mensaje_separado[0] + '#' + i.strip() + " " +
mensaje_separado[1]
    print("Con hashtags: ", mensaje)
    return mensaje

def actualizar_trending_topics(hashtag):
    global conteos_verano
    global conteos_iic1103
    global conteos_programar

    if hashtag.lower() in lista_topicos:
```

```
        if hashtag.lower() == "verano":
            conteos_verano += 1
        elif hashtag.lower() == "iic1103":
            conteos_tav += 1
        elif hashtag.lower() == "programar":
            conteos_programar += 1

#Principal
print(twittear("me encanta el iic1103, el #verano y #programar", False))
print("Verano: ", conteos_verano)
print("Programar: ", conteos_programar)
```

Pregunta 4

La administración de información académica necesita un programa que permita obtener los promedios y las estadísticas de cursos de la universidad. Por este motivo se te pide que hagas lo siguiente:

a. Implementa la función de nombre ***calcularNotaPromedio*** que reciba como parámetro las notas que un alumno obtiene en un curso, y que calcule y retorne el promedio ponderado de estas notas. Para esto las notas de un alumno están definidas como una lista de la forma: [(nota11,ponderacion11), (nota12,ponderacion12), ...]. Considera que cuando un alumno no tiene todas sus notas el promedio no se puede calcular y por ende debes dejar la nota en categoría de *nota pendiente* (np).

   Por ejemplo, si un alumno tiene las siguientes notas en el curso [(**7**,**0.3**), (**5**,**0.4**), (**4.8**,**0.3**)] entonces indica que alumno tiene 3 notas, un 7 con ponderación 30%, un 5 con ponderación de un 40% y un 4.8 con ponderación de un 30%. Para este caso, el resultado que se obtiene al invocar la función ***calcularNotaPromedio*** es 5.5

   Solución:

```
def calcularNotaPromedio(notas_alumno):
    suma_ponderadores = 0
    nota = 0
    for i in notas_alumno:
        suma_ponderadores += i[1]
        nota += i[0]*i[1]

    if int(suma_ponderadores) == 1:
        return nota
    else:
        return "np"
```

b. Los desafíos de la administración de información académica son aun mayores, y para una mejor gestión necesitan obtener las estadísticas de notas de los cursos. Por eso te piden que implementes la función de nombre ***calcularEstadisticas*** que recibe como parámetro la información de todas las notas de un curso y que devuelva el promedio de las notas (de los alumnos de ese curso), la nota máxima y la nota mínima en el curso.

Considera que la información de un curso esta definida por un diccionario que relaciona el número de alumno con su lista de notas, en el análisis no se incluyen las notas con promedios con nota pendiente (np). Por ejemplo, el curso IIC1103 está definido por:

```
iic1103 = {'A001':[(7.0,0.3),(5.0,0.4),(4.8,0.3)],
        'A002':[(4.0,0.1),(5.4,0.7),(1.5,0.2)],
        'A003':[(4.0,0.1),(5.5,0.7),(2.0,0.2)],
        'A004':[(4.0,0.1),(5.2,0.7)],
        }
```

Y las estadísticas del curso iic1103 son (4.89, 5.54, 4.48). El resultado de la función puede ser a través de cualquiera de la estructuras estudiadas (listas, tuplas o diccionarios). Notar que es posible que ningún alumno tenga sus notas completas.

Solución

```python
def calcularEstadisticas(datos_curso):
    nota_min = 7
    nota_prom = 0
    nota_max = 0
    count = 0

    for i in datos_curso:
        auxiliar = calcularNotaPromedio(datos_curso[i])

        if auxiliar != "np":

            if auxiliar < nota_min:
                nota_min = auxiliar

            if auxiliar > nota_max:
                nota_max = auxiliar

            count += 1
            nota_prom += auxiliar
        else:
            print(i,"NP")
    if count > 0:
        return [nota_min, nota_prom/count, nota_max]
    else:
        return "Ningún alumno tiene sus notas"
```

Éxito !!!