FACULTAD DE INGENIERÍA
FUNDAMENTOS DE COMPUTACION Y PROGRAMACION

PRUEBA ACUMULATIVA

ASPECTOS GENERALES DE LA PRUEBA

- Lea atentamente la prueba y las instrucciones antes de comenzar a desarrollarla.
- Queda prohibido hablar con los compañeros(as) durante el desarrollo de la PEP.
- La PEP contiene 2 preguntas de desarrollo, con un total de **45** puntos y una exigencia del **60%**
- Tiene un límite de tiempo de **90** minutos para responder.
- El equipo docente tiene la prohibición de responder consultas.
- El/La estudiante que se sorprenda en actos deshonestos será calificado con la nota mínima.
- Los elementos tecnológicos deben permanecer apagados y guardados. Queda absolutamente prohibido el uso todo elemento tecnológico. Su uso puede significar la nota mínima o sanciones mayores.
- El alumno deberá identificarse con su Cédula de Identidad.
- Sobre el escritorio sólo podrá existir lápiz (obligatorio) y goma/lápiz corrector (opcional).
- Complete sus datos personales antes de comenzar la evaluación.
- Considere que la evaluación contempla el código, los comentarios y el seguimiento de las buenas prácticas de programación.
- Responda cada pregunta, a continuación de su enunciado, en el espacio que se le entrega para ello.

NOMBRE	RUT	SECCIÓN	

1. **(15 puntos)** Construya un programa en Python que para un archivo de entrada “texto.txt”, encuentre e informe al usuario de la palabra más larga, es decir, aquella que tiene la mayor cantidad de caracteres y la cantidad de caracteres que esta tiene.

   Considere que las palabras en el archivo sólo están separadas por espacio y saltos de línea, y en caso de que exista más de una palabra de largo máximo, basta con que se informe una de ellas.

   Por ejemplo, para el archivo texto.txt, la salida sería la indicada:

texto.txt
Hello Today I had a traffic accident It was not my fault I am fine my family is fine And thanks for everything

Salida
La palabra más larga es: everything con 10 caracteres

```
# BLOQUE DE DEFINICIONES

# Función que lee un archivo de texto
# Entrada: Nombre del archivo (string)
# Salida: Contenido del archivo (lista de strings)
def leerArchivo(nombreArchivo):
    # Se abre el archivo en modo de lectura
    archivo = open(nombreArchivo, 'r')
    # Se obtiene el contenido del archivo
    contenido = archivo.readlines()
    # Se cierra el archivo
    archivo.close()
    # Se entrega el contenido
    return contenido

# Función que separa las palabras de cada línea
# Entrada: Contenido del archivo (lista de strings)
# Salida: Lista de palabras (lista de strings)
def separarPalabras(contenido):
    # Se declara una lista vacía para almacenar las palabras
    listaPalabras = []
    # Para cada línea en la lista de strings
    for linea in contenido :
        # Se eliminan los saltos de línea
        linea = linea.strip('\n')
        # Se realiza un split por el caracter espacio
        # para obtener cada palabra por separado
        linea = linea.split(" ")
        # Se concatena la lista de palabras obtenidas en la línea
        # con la lista de palabras global
        listaPalabras = listaPalabras + linea
    # Se retorna la lista de palabras
    return listaPalabras

# Función que encuentra la palabra con más caracteres
# dentro de una lista de palabras
# Entrada: Lista de palabras (lista de strings)
# Salida: Palabra más larga (string)
def encontrarMayor(listaPalabras):
    # Se declara un string vacío para almacenar la palabra
    # de mayor tamaño
    mayor = ""
    # Para cada palabra en la lista de palabras
    for palabra in listaPalabras :
        # Si el largo de la palabra a revisar es mayor
        # que el largo del valor almacenado en mayor
        if len(palabra) > len(mayor) :
            # La palabra revisada es guardada en mayor
            mayor = palabra
    # Al terminar el ciclo en la variable mayor tengo
```

```
    # la palabra de mayor tamaño
    return mayor

# BLOQUE PRINCIPAL
# ENTRADA
# Se lee el archivo, invocando la función leerArchivo
contenidoArchivo = leerArchivo("texto.txt")

# PROCESAMIENTO
# Se invoca al proceso de separación de palabras
palabras = separarPalabras(contenidoArchivo)
# Se invoca al proceso para encontrar la palabra de mayor
# tamaño
palabraMasLarga = encontrarMayor(palabras)
# Se determina el largo de la palabra utilizando la función len
numeroCaracteres = len(palabraMasLarga)
# SALIDA
# Se imprime la salida solicitada
print "La palabra más larga es: ", palabraMasLarga
print "con", numeroCaracteres, "caracteres"
```

2. **(15 puntos)** De acuerdo a la suma de sus divisores, los números pueden clasificarse como deficientes, perfectos o abundantes.

- Un número deficiente es un número en el cuál la suma de sus divisores es menor al doble del número, por ejemplo 14 es un número deficiente pues:
  `1 + 7 + 2 + 14 = 24 < 2 * 14 = 28`
- Un número perfecto es un número en el cuál la suma de sus divisores es igual al doble del número, por ejemplo 28 es un número perfecto pues:
  `1 + 2 + 4 +7 + 14 + 28 =  56  = 28 * 2`
- Un número abundante es un número en el cual la suma de sus divisores es mayor al doble del número, por ejemplo 12 es un número abundante pues:
  `1 + 2 + 3 + 4 + 6 + 12 = 28 > 12 * 2 = 24`

Construya un programa en Python, que indique si un número es deficiente, perfecto o abundante.

```
# BLOQUE DE DEFINICIONES
# Función que encuentra los divisores de un número
# Entrada: Número entero positivo
# Salida: Lista de enteros
def encontrarDivisores(numero):
    # Se declara una lista vacía para
    # almacenar los divisores del número
    divisores = []
    # Se declara un iterador y se inicializa en 1
    # para evitar la división por cero
    i = 1
```

```
    # Mientras el valor de i sea menor o igual que el número
    while i <= numero :
        # Si i divide exactamente al número
        if numero % i == 0 :
            # Se agrega a la lista de divisores
            divisores.append(i)
        # Se incrementa el valor de i
        i = i + 1
    # Se retorna la lista de divisores
    return divisores

# BLOQUE PRINCIPAL

# ENTRADA
# Se solicita el ingreso del número
numeroIngresado = input("Ingrese un número entero positivo: ")

# PROCESAMIENTO
# Se invoca a la función encontrar divisores para obtener la lista
divisoresDelNumero = encontrarDivisores(numeroIngresado)
# Se declara una variable para almacenar la suma
sumaDivisores = 0
# Para cada elemento en la lista divisoresDelNumero
for elemento in divisoresDelNumero :
    # Se acumula en la variable sumaDivisores
    sumaDivisores = sumaDivisores + elemento
# Se resta el total de la suma de divisores al doble del número ingresado
# para generar la condición a revisar
condicionARevisar =  numeroIngresado * 2 - sumaDivisores

# SALIDA
# Si el cálculo resulta negativo
if condicionARevisar < 0 :
    # Se informa al usuario que el número es deficiente
    print "El número", numeroIngresado, "es deficiente"
# Si el cálculo resulta positivo
if condicionARevisar > 0 :
    # Se informa al usuario que el número es abundante
    print "El número", numeroIngresado, "es abundante"
# Si el cálculo es exactamente cero
if condicionARevisar == 0 :
    # Se informa al usuario que el número es perfecto
    print "El número", numeroIngresado, "es perfecto"
```

---

3. **(15 puntos)** Una limaçon o caracol de Pascal es la coincoide de una circunferencia que pasa por el polo, y se define paramétricamente como:

$$r = r_0 + \cos\theta$$
$$x = r\ \cos\theta$$
$$y = r\ \sin\theta$$

Cuando $r_0 = 1$, la curva es llamada cardioide, al variar este valor la curva cambia en su forma. Use esta definición para graficar, en un mismo gráfico, la forma de una limaçon para:

- $r_0 = 0.5$ en rojo
- $r_0 = 1.0$ en verde
- $r_0 = 1.5$ en azul

Para el ángulo θ en el intervalo $[-\pi, \pi]$, para su desarrollo considere que:

- El módulo numpy contiene las funciones trigonométricas `sin(array)` y `cos(array)` que calculan el seno y coseno de un vector de ángulos dados en radianes.
- El módulo math contiene las funciones trigonométricas `sin(float)` y `cos(float)` que calculan el seno y coseno de un valor numérico en radianes.
- Tanto el módulo math como el módulo numpy permiten la importación del valor de PI

```
# IMPORTACIÓN DE MODULOS

# Se importa el módulo numpy
import numpy
# Se importa el módulo matplotlib.pyplot
import matplotlib.pyplot as plotter

# CONSTANTES
# Se declaran los tres casos a graficar
VALOR_R0_CASO_1 = 0.5
VALOR_R0_CASO_2 = 1.0
VALOR_R0_CASO_3 = 1.5

# Se importa desde numpy el valor de PI
PI = numpy.pi
# Se declara el salto que tendrá el vector de ángulos
SALTO = 0.0001

# ENTRADAS
# No existen entradas, pues todos los valores entregados
# son constantes

# PROCESO

# Se genera el vector para el ángulo en el intervalo
# de -PI a PI
angulo = numpy.arange(-PI, PI + SALTO, SALTO)
```

```
# Para el caso 1
# Se determina el valor de r para r0 = 0.5
vectorR0Caso1 = VALOR_R0_CASO_1 + numpy.cos(angulo)
# Se genera el vector para el eje X
vectorXCaso1 = vectorR0Caso1 * numpy.cos(angulo)
# Se genera el vector para el eje y
vectorYCaso1 = vectorR0Caso1 * numpy.sin(angulo)

# Para el caso 2
# Se determina el valor de r para r0 = 1.0
vectorR0Caso2 = VALOR_R0_CASO_2 + numpy.cos(angulo)
# Se genera el vector para el eje X
vectorXCaso2 = vectorR0Caso2 * numpy.cos(angulo)
# Se genera el vector para el eje y
vectorYCaso2 = vectorR0Caso2 * numpy.sin(angulo)

# Para el caso 3
# Se determina el valor de r para r0 = 1.5
vectorR0Caso3 = VALOR_R0_CASO_3 + numpy.cos(angulo)
# Se genera el vector para el eje X
vectorXCaso3 = vectorR0Caso3 * numpy.cos(angulo)
# Se genera el vector para el eje y
vectorYCaso3 = vectorR0Caso3 * numpy.sin(angulo)

#SALIDAS

# Se grafica la primera curva con r0 = 0.5
curva1 = plotter.plot(vectorXCaso1, vectorYCaso1)
# Se setea el color de la línea en rojo
plotter.setp(curva1, 'color', 'r')

# Se grafica la segunda curva con r0 = 1.0
curva2 = plotter.plot(vectorXCaso2, vectorYCaso2)
# Se setea el color de la línea en verde
plotter.setp(curva1, 'color', 'g')

# Se grafica la tercera curva con r0 = 1.5
curva3 = plotter.plot(vectorXCaso3, vectorYCaso3)
# Se setea el color de la línea en azul
plotter.setp(curva1, 'color', 'b')

# Se le agrega un título al gráfico
plotter.title("Grafico de caracoles de pascal")

# Se le agrega un rótulo al eje X
plotter.xlabel("r cos(angulo)")
# Se le agrega un rótulo al eje Y
plotter.ylabel("r sin(angulo)")
# Se muestra el gráfico
plotter.show()
```