UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Intro. a Ciencia de la Computación
FUNCIONES
Pregrado
2019-II
Ciencia de la Computación
Laboratorio

Las siguientes preguntas se recogen de prácticas pasadas y están acompañadas de un indicador de dificultad que va desde el 1 al 5.

1. (Nivel 1) Deberás crear una función que se llame *"maravilla"*. Esta función recibirá 3 parámetros, y tendrá las siguientes características:

   - El primer parámetro sera un carácter, el cual puede ser alguno de estos símbolos:
     "+" (más)
     "-" (menos)
     "*" (por)
     "/" (entre)
   - El segundo y el tecer parámetros serán dos números enteros .
   - La función devolverá el resultado de la operación enviada.
   - Si el primer parámetro fuera "/" ("entre") y el segundo número un cero, deberá imprimir un mensaje de error y no devolverá nada.

2. (Nivel 1) Implemente una función que reciba un número como parámetro y haga las sumatorias de la siguiente forma:

   - Si el número es par la función debe calcular y retornar la suma de todos los números pares desde el 0 hasta el número pasado como parámetro.
   - Si el número es impar entonces la función debe calcular y retornar la suma de todos los números impares desde el 0 hasta el número pasado como parámetro.
   - Si el número es 0 o no se le pasa ningun párametro, debe retornar el valor de 0.

   Luego implemente un programa que utilize la función anterior, solicitando al usuario que ingrese un número e imprimiendo el resultado.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número: 5
Resultado: 9
```

```
Ingrese un número: 6
Resultado: 12
```

```
Ingrese un número: 9
Resultado: 25
```

3. (Nivel 1) Dada una lista de números construya una función que extraiga una secuencia de números dada dos posiciones. Presente el resultado como la suma de esos números extraídos. Los números ingresados se incorporan como una lista en una sola línea.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 3 4 5 6 7 9 1 2 3 4
pos1: 3
pos2: 5

Output: 18
```

```
Input: 3 4 1 2 7 9 1 2 3 4
pos1: 5
pos2: 8

Output: 19
```

4. (Nivel 1) Implemente una función que reciba un número entero como parámetro y retorne el factorial de dicho número. En caso no se envíe ningún valor al momento de invocarla, deberá calcular el factorial de 10. Algunos ejemplos de diálogo de este programa serían:

```
Input: 3
Output: 6
```

```
Input: 5
Output: 120
```

```
Input:
Output: 3628800
```

5. (Nivel 1) Dada una matriz de valores:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

   Se define la determinante como $D = ad - bc$

   Construya una función en Python para obtener la determinante:

   - Escriba la función como parte de un archivo módulo aparte: ejemplo determinante.py.

- Escriba un programa main.py en donde se lea los valores *a, b, c, d* y devuelva el resultado de su determinante.
- No olvide invocar con import el módulo de su función.

Algunos ejemplos de ejecución del programa:

```
input:
3
4
5
6
output:
-2
```

```
input:
10
3
7
8
output:
59
```

6. (Nivel 1) Se tiene 3 cañitas, posiblemente de diferentes longitudes, puede o no ser posible colocarlos de manera que formen un triángulo cuando sus extremos se toquen. Por ejemplo:

   - si todas las cañitas tienen una longitud **de 6 cm** entonces uno puede construir fácilmente un triángulo equilátero.
   - si una cañita **mide 6 cm** de largo, mientras que el los otros dos son cada uno de **2 cm**, entonces no se puede formar un triángulo.

   En general, si alguna longitud es mayor o igual que la suma de las otras dos, entonces las longitudes no se puede utilizar para formar un triángulo. El caso contrario si se puede formar un triángulo.

   Escribe una función que determine si tres longitudes pueden formar un triángulo o no.

   Construya la función en Python para obtener ese resultado:

   - La función tomará 3 parámetros y devolverá un resultado booleano.
   - Escriba la función como parte de un archivo módulo aparte.
   - Escriba un programa en donde se invoque a la función indicada, lea un valor y devuelva el resultado.

   Algunos ejemplos de ejecución del programa:

```
3 4 5
True
```

```
2 6 2
False
```

7. (Nivel 1) Desarrollar un programa en Python que permita ingresar N códigos. Debe validar que no se ingrese un código duplicado, para ello debe crear obligatoriamente una función que valide la duplicidad. Ejemplo de diálogo de este programa:

```
Ingrese total de codigos: 4
Codigo 1: 125
Codigo 2: 100
Codigo 3: 125
ERROR, codigo duplicado
Codigo 3: 63
Codigo 4: 100
```

8. (Nivel 1) Implemente una función que reciba una lista con números y devuelva el mayor. También implemente una función que reciba una lista con números y devuelva el menor. Luego implemente un programa que solicite 5 números, los ponga en una lista y llame a esas dos funciones. Finalmente imprime el promedio entre el mayor y el menor.

   Un ejemplo de este programa sería:

```
Ingrese el nro 1: 5
Ingrese el nro 2: 7
Ingrese el nro 3: 10
Ingrese el nro 4: 2
Ingrese el nro 5: 100
51
```

9. (Nivel 1) Usted es jurado calificador de un concurso de proyectos tecnológicos. Para ello, debe evaluar el performance de los equipos participantes en función a diversos criterios de calificación y en función de eso es que se definirá el ganador del concurso. Como el proceso hasta ahora ha sido netamente manual, el equipo organizador del concurso ha convenido que debe ahora ser un proceso automatizado.

   Por ello, se le ha encargado diseñar e implementar un programa que:

   - Reciba un número N de equipos para ser evaluados.
   - Implementar una módulo que contenga una función para calificar al equipo.
     - Esta función debe devolver la calificación en base a:
       * Existen cuatro criterios de calificación. Cada criterio (a,b,c,d) puede ser calificado del 1 al 3.

- En ese mismo módulo se debe implementar una función que reciba la calificación de cada equipo y devuelva la calificación más alta.
- Deberá imprimir la calificación más alta correspondiente al equipo ganador.

Caso de prueba:

```
Grupos: 4
Resultados Grupo 1 = [3,1,3,2]
Resultados Grupo 2 = [3,2,3,2]
Resultados Grupo 3 = [3,3,1,2]
Resultados Grupo 4 = [3,3,3,3]
Calificacion ganador = 12
```

10. (Nivel 2) Implemente una función que reciba un número entero como parámetro y retorne la sumatoria de dicho número. En caso no se envíe ningún valor al momento de invocarla, deberá calcular la sumatoria de 10.

    Algunos ejemplos de diálogo de este programa serían:

```
Input: 5
Output: 15
```

```
Input: 10
Output: 55
```

11. (Nivel 2) Implementar un algoritmo que recibe un número entero menor a 9 en notación decimal y lo convierte a binario.

    - Debe implementarse una función. En caso el número pasado como argumento no cumpla con el requisito imprime un mensaje "Por favor inténtelo nuevamente". En caso no se ingrese un número para la conversión, por defecto se calcula la conversión de 8.
    - Cada dígito de la conversión es almacenado en una lista.
    - La conversión es impresa de forma inversa, esto facilita la solución. Por ejemplo, 6 en binario es 110, pero se almacenaría como [0, 1, 1].
    - **TIP:** usar la conversión por división y *while*.

    Algunos ejemplos de diálogo de este programa serían:

```
Input: conversion()
Output: [0, 0, 0, 1]
```

```
Input: conversion(3)
Output: [1, 1]
```

```
Input: conversion(9)
Output: Por favor inténtelo nuevamente
```

```
Input: conversion(8)
Output: [0, 0, 0, 1]
```

12. (Nivel 2) Diseñe e implemente un algoritmo que calcule la suma de cubos de los dígitos de un número divisible por 3. Para ello, ingrese un número entero y determine si es divisible por 3. En caso no lo sea, pida otro número que sea divisible por 3 hasta que se ingrese uno correcto. Luego calcule la suma de cubos de los dígitos del número ingresado. A continuación, calcule la suma de cubos de los dígitos del resultado anterior. Continúe el procedimiento hasta que el cálculo sea igual a 153.
¿Qué observa en el resultado?
E.g., para el número 333:

```
Ingrese un numero: 333
la suma de cubos de 333 es: 81
la suma de cubos de 81 es: 513
la suma de cubos de 513 es: 153
la suma de cubos de 153 es: 153
```

13. (Nivel 2) Elabore un programa que contenga una función llamada *validarango*.

    Esta función recibirá como parámetros los siguientes valores:

    - Rango inicio (*ri*): Es un número entero que representa el valor inicial del rango.
    - Rango fin (*rf*): Es un número entero que representa el valor final del rango.
    - Número (*numero*): Número entero para el cual, la función validará si su valor se encuentra entre *ri* y *rf*.

    La función retorna `True` si el número es validado y `False` en caso contrario.

    En el programa principal el usuario ingresará la cantidad de números válidos requeridos para calcular el promedio. Un número es válido si está dentro del rango especificado. Luego de que el usuario ingresó todos los números válidos requeridos, el programa mostrará el promedio de estos.

    Algunos ejemplos de diálogo de este programa serían:

```
Ingrese cantidad de numeros validos esperados: 3
Ingrese rango de inicio: 100
Ingrese rango de fin: 200
Ingrese numero 1:30
Ingrese numero dentro del rango: [100 - 200]
Ingrese numero 1:10
Ingrese numero dentro del rango: [100 - 200]
```

```
Ingrese numero 1:150
Ingrese numero 2:45
Ingrese numero dentro del rango: [100 - 200]
Ingrese numero 2:160
Ingrese numero 3:180
El promedio es: 163.33333333333334
```

```
Ingrese cantidad de numeros validos esperados: 4
Ingrese rango de inicio: 60
Ingrese rango de fin: 80
Ingrese numero 1:10
Ingrese numero dentro del rango: [60 - 80]
Ingrese numero 1:65
Ingrese numero 2:76
Ingrese numero 3:80
Ingrese numero 4:60
El promedio es: 70.25
```

14. (Nivel 2) Rosa es estudiante del colegio Nuestra Señora de la Merced de San Juan de Lurigancho, y en su curso de aritmética hicieron el tema de divisibilidad. El profesor del curso mostró los criterios de divisibilidad del número 11 que permiten averiguar si un número es divisible por 11 de una forma sencilla, sin utilizar módulo.

    El criterio de divisibilidad es el siguiente:

    - Un número es divisible entre 11 cuando la **suma** de los números que ocupan la posición par **menos** la **suma** de los números que ocupan la posición impar es igual a **cero**.

    La aplicación del criterio de divisibilidad para el número `455433`, es:

    - Para saber si `455433` es divisible entre 11, identificamos cuáles son las cifras que ocupan las posiciones pares y las que ocupan las posiciones impares.
    - Posición par: 4, 5 y 3. Los sumamos: $4 + 5 + 3 = 12$
    - Posición impar: 5, 4 y 3. Los sumamos: $5 + 4 + 3 = 12$
    - Restando: $12 - 12 = 0$, por lo tanto el número `455433` SI es divisible entre 11.

    Considere que todos los números que se ingresarán tendrán un tamaño par.

    Escribe un programa que permita al usuario ingresar un número decimal, y el programa debe averiguar si el número es divisible por 11, y al final debe imprimir "`El número SI es divisible por 11`" cuando el número sea divisible por 11 y "`El número NO es divisible por 11`" en caso contrario.

    Para esto tu programa debe implementar y usar la siguiente función:

- `isDivisible11` recibe como parámetro un `número` y retorna `SI` en el caso que sea divisible entre 11 y `NO` en caso contrario.

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número: 14587629
El número NO es divisible por 11
```

```
Ingrese un número: 783475
El número SI es divisible por 11
```

15. (Nivel 2) Diseñe e implemente una función *extrae_intervalo* que reciba tres parámetros: una lista y dos enteros opcionales, el primero de ellos representa el límite inferior y el segundo el superior. La función debe retornar una lista que contenga aquellos elementos que se encuentran dentro del intervalo formado por el límite inferior y superior (contando los límites). En caso de que no se envien los límites al momento de invocar la función deberá considerarse 0 como límite inferior y 100 como límite superior.

```
lista = [1, 80, 15, 30, 25, 90, 110, 105, 180, 200, 10]
print(extrae_intervalo(lista, 10, 70)
15 30 25 10
print(extrae_intervalo(lista)
1 80 15 30 25 10
```

16. (Nivel 2) Diseñe e implemente una función *puntaje* que reciba tres parámetros: una lista que contiene unicamente tres valores: G (ganado), E (empatado) y P (perdido). Los dos parámetros restantes serán dos enteros que representen el puntaje a asignar en caso de partido ganado y empatado (el partido perdido siempre valdrá cero). En caso estos dos parámetros no se envíen al momento de invocar a la función se deberá considerar tres puntos por partido ganado y un punto por partido empatado. Puede asumir que los únicos valores que vendrán en la lista son "G", "E" o "P".

```
lista = ["G", "G", "P", "P", "E", "G"]
print(puntaje(lista, 5, 2))
17
print(puntaje(lista))
10
```

17. (Nivel 3) Implemente una función que reciba un número como parámetro, y haga los cálculos de la siguiente forma:

    - Si el número es múltiplo de 3 entonces la función debe calcular y retornar la suma de todos los números múltiplos de 3 desde el 0 hasta el número pasado como parámetro.
    - Si el número es múltiplo de 4 entonces la función debe calcular y retornar la suma de todos los números múltiplos de 4 desde el 0 hasta el número pasado como parámetro.

- Si el número es múltiplo de 3 y 4 entonces la función debe retornar la suma de los dos cálculos anteriores
- Si el número no es múltiplo de 3 ni de 4, o no se le pasa ningun párametro, debe retornar el valor de 0.

Luego implemente un programa que utilize la función anterior, solicitando al usuario que ingrese un número e imprimiendo el resultado.

Algunos ejemplos de diálogo de este programa serían:

```
Input: 16
Output: 40
```

```
Input: 9
Output: 18
```

```
Input: 12
Output: 54
```

```
Input: 17
Output: 0
```

18. (Nivel 3) Crea una función que reciba una frase y devuelva en una lista todos los caracteres de la frase, pero sin repetidos. Crea una función que reciba una frase e indique cuántos caracteres están repetidos. El programa principal usará ambas funciones y devolverá la lista creada por la función y el número de caracteres repetidos. El espacio en blanco no será considerado como un carácter.

    Un ejemplo de este programa sería:

```
Ingresa la frase: soy el chef mas acebichado
["s","o","y","e","l","c","h","f","m","a","b","i","d"]
6
```

19. (Nivel 3) Usted se encuentra trabajando en una empresa de educación y ha sido encargado con la misión de calcular la rentabilidad que un taller que están ofreciendo; esto le permitirá tomar decisiones sobre el negocio. Para ello, debe entender bien la dinámica del negocio y, una vez logrado, debe automatizar el proceso de cálculo en un programa en python.

    Diseñe e implemente un programa que obedezca la siguiente lógica:

    - El programa debe recibir la cantidad de estudiantes matriculados en el taller.
    - El programa debe llamar a una función llamada *costos_fijos* que devuelva el costo fijo total sumando todos los costos fijos:
        - Alquiler: $ 160

  - Publicidad: $ 50
- El programa debe llamar a una función llamada *costos_variables* que devuelva el costo variable total:
  - Hasta los 10 alumnos el profesor cobra: $ 10.00 por hora
  - Desde los 11 hasta los 15 alumnos el profesor cobra: $ 13.00 por hora
  - Desde los 16 a 20 el profesor cobra: $. 17.00
  - A partir de 21 alumnos cobra: $ 20.00
  - Todos los talleres duran 16 horas.
- El programa debe llamar a una función llamada *evaluo_renta* que:
  - Debe calcular los ingresos en función al precio de venta ($ 80.00 por estudiante).
  - Restarle los costos totales (suma de los costos fijos y los costos variables)
  - Devolver 'es rentable' si el resultado es mayor a 0 y 'no es rentable' si el resultado es negativo.
- El programa debe usar un solo módulo donde estén creadas las funciones solicitadas.
- El resultado de la función *evaluo_renta* debe ser impreso al final del programa.

Algunos casos de prueba:

```
Ingrese cantidad de estudiantes: 20
Resultado: es rentable
```

```
Ingrese cantidad de estudiantes: 3
Resultado: no es rentable
```

20. (Nivel 3) Diseñar y crear una función que se llamará **contar_omitidos** que tome una lista de cualquier tamaño (de 2 a más valores) cuyo contenido sean letras ordenadas en forma creciente (de 'a' hasta 'z') continuas o separadas por 1 o más caracteres. La función deberá contar cuantas letras han sido omitidas para que la secuencia de letras sea continua.

```
print(contar_omitidos(['a', 'd', 'e', 'i']))
5
```

```
print(contar_omitidos(['a', 'f', 'p']))
13
```

```
print(contar_omitidos(['f', 'p']))
9
```

```
print(contar_omitidos(['b', 'c', 'd', 'e', 'f', 'g']))
0
```

21. (Nivel 4) Diseñe e implemente una función que se llame **contar_omitidos** que reciba como parámetro una lista de caracteres ordenados de forma creciente (de 'a' hasta 'z') continuas o separadas por 1 o más caracteres. La función deberá contar cuantas letras han sido omitidas para que la secuencia de letras sea continua.

    Algunos ejemplos de diálogo de este programa serían:

```
print(contar_omitidos (['a', 'd', 'e', 'i']))
omitidos: 5
```

```
print(contar_omitidos (['a', 'f', 'p']))
omitidos: 13
```

```
print(contar_omitidos (['f', 'p']))
omitidos: 9
```

22. (Nivel 4) Dada la siguiente función matemática:

$$F = \frac{(\cos(t) + \cos(2t) + \cos(3t) + ... + \cos(Nt))}{N} \tag{1}$$

    El valor de t puede estar solo entre -10 y 10. El valor entero recibido deberá convertirse a radianes con la siguiente expresión $t = math.radians(grado)$ Construya la función en Python para obtener ese resultado:

    - Escriba la función como parte de un archivo módulo aparte.
    - Escriba un programa en donde se invoque a la función indicada, lea N y el *grado* y devuelva el resultado.
    - Utilice import math para acceder a la función coseno.

    Algunos ejemplos de ejecución del programa:

```
20
1
0.9782829790376819
```

```
5
2
0.9933105188801943
False
```

23. (Nivel 4) Realizar un programa que nos permita realizar todas las funciones de una calculadora básica utilizando funciones, es decir, en el programa principal debe indicarnos qué tipo de operación deseamos realizar: *suma, resta, multiplicación, división y resultados anteriores*. Si el usuario ingresó una opción que no existe, entonces nuevamente debe volver a preguntarle que ingrese una opción correcta. El usuario debe ingresar la

operación a realizar y los valores; en el caso que el usuario ingrese *resultados anteriores*, se debe imprimir la lista con los resultados anteriores; en el caso que sea una operación, el programa debe llamar funciones que realicen el cálculo deseado regresando el valor a la función principal, la cual va a imprimir el resultado y llamar a la función guardar para que se guarde el resultado en la lista. Deben existir 2 archivos: **principal.py, funciones.py**. En el principal.py se le pide **SIEMPRE** al usuario qué tipo de operación desea realizar, ingresar los datos e imprime el resultado. En funciones.py. es donde estarán las operaciones. Algunos ejemplos de diálogo de este programa serían:

```
Ingresan: suma 3 4
Salida: 7
```

```
Ingresan: multiplicacion 14 2
Salida: 28
```

```
Ingresan: resultados anteriores
Salida: [7, 28]
```

24. (Nivel 4) Erika Fernandez Bodas - Eventos, tiene más de 5 años en el rubro de eventos. Ofrece servicios de calidad, deliciosos catering y una decoración personalizada. Realizan todo tipo de eventos como coffee breaks, aniversarios, fiestas de fin de año, bodas, celebraciones de cumpleaños, despedidas de solteras/os, comidas de empresa, cenas temáticas y etc.

    Todos los eventos de Erika están calendarizados, pero a ella le interesaría tener la cantidad de días que falta para cada evento.

    Procedimiento:
    Considerando que hoy es 17/01/2019 y la fecha del evento es: 16/05/2019, calculamos los días por mes:

    - Enero: 14 días
    - Febrero: 28 día
    - Marzo: 31 días
    - Abril: 30 días
    - Mayo: 16 días

    Total días que faltan: 14 + 28 + 31 + 30 + 16 = **119 días**.

    Ayudemos a Erika a obtener la cantidad de días que falta para un evento. Escribe un programa que permita al usuario ingresar el día, mes y año de una fecha, y el programa debe obtener la cantidad de días, y al final debe imprimir la cantidad de días.

    Para esto tu programa debe implementar y usar la siguiente función:

    - `obtenerDias` recibe como parámetro `día, mes y año` y retorna `cantidad de días`, considere que hoy es 17 de enero del 2019.

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese día: 16
Ingrese mes: 5
Ingrese año: 2019
Dias faltantes:  119
```

```
Ingrese día: 20
Ingrese mes: 9
Ingrese año: 2019
Dias faltantes:  246
```

25. (Nivel 4) Para dar solución a este problema se requiere lo siguiente:

    - Crea una función que reciba una cadena de caracteres y te devuelva el carácter que más se repite.
    - Crea un programa que pida al usuario 3 frases.
    - El programa mostrará en pantalla una cadena formada por lo caracteres más repetidos de cada una de las tres frases

    Algunos ejemplos de diálogo de este programa serían:

```
Ingrese la frase 1: mi mama me mima
Ingrese la frase 2: ojos que no ven, ciego es
Ingrese la frase 3: alabama
Resultado: moa
```

26. (Nivel 4) Dada una tabla de contingencia en donde se ha registrado la ocurrencia de tres enfermedades para una muestra aleatoria de 1000 individuos por cada ciudad. Se le pide elaborar un programa que permita calcular la *probabilidad marginal* de cada enfermedad. Esta probabilidad se obtiene sumando las ocurrencias de dicha enfermedad en cada ciudad y el resultado es dividido por la suma total de la tabla, es decir: sea la matriz $M[n][m]$, la probabilidad marginal de la columna c se obtiene:

$$PM(c) = \frac{\sum_{i=0}^{n-1} M[i][c]}{\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} M[i][j]}$$

    Para el desarrollo considerar dos funciones: una para calcular la suma total de elementos de la tabla y la otra función para calcular la suma de elementos de una columna. La matriz de datos y la lista de enfermedades debe ser definido dentro del código.

    Ejemplo: en mi código tengo la siguiente matriz de datos.

```
Ingrese enfermedad: Gripe
La probabilidad marginal es: 0.45
```

	Gripe	Influenza	Cólera
Ciudad 1	200	460	340
Ciudad 2	620	180	200
Ciudad 3	530	250	220

27. (Nivel 5) Escribir un programa que tenga como entrada dos números enteros positivos **A** y **B** donde (**A** <**B**) y que tenga como salida todos los números primos que existen entre **A** y **B**.

    Considerar los siguientes puntos:

    - Si el segundo número ingresado es menor que el primer número ingresado entonces el programa imprime el siguiente mensaje *"El segundo número ingresado tiene que ser mayor que el primer número!"* y se seguirá pidiendo los números **A** y **B** hasta que se cumpla esta condición.
    - Crear un módulo con nombre *"funciones.py"* e implementar la función con nombre *"es_primo"* para saber si un número es primo o no.
    - Crear un módulo *"main.py"* para importar el modulo *funciones* y proceder a implementar el algoritmo de encontrar los números primos entre **A** y **B**.
    - El resultado debe mostrar a los números primos horizontalmente espaciados.

    Ejemplo del programa:

```
python main.py
Ingrese el primer numero: 2
Ingrese el segundo numero: 1
El segundo numero ingresado tiene que ser mayor que el
   primer numero!
Ingrese el primer numero:45
Ingrese el segundo numero: 10
El segundo numero ingresado tiene que ser mayor que el
   primer numero!
Ingrese el primer numero:2
Ingrese el segundo numero: 100
Todos los numeros primos que existen entre 2 y 100 son:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
   79 83 89 97
```

28. (Nivel 5) El sistema hexadecimal es el sistema de numeración posicional que tiene como base el 16. Su uso actual está muy vinculado a la informática y ciencias de la computación donde las operaciones de la CPU suelen usar el byte u octeto como unidad básica de memoria.

    Dado que el sistema usual de numeración es de base decimal y, por ello, sólo se dispone de diez dígitos, se adoptó la convención de usar las seis primeras letras del alfabeto

latino para suplir los dígitos que hacen falta. A continuación se muestra la tabla de equivalencia.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

El método para convertir un número decimal a hexadecimal es:

- Dividimos el número entre 16.
- Si el cociente es mayor o igual que 16, lo dividimos entre 16.
- Continuamos así hasta obtener un cociente menor que 16.
- Concatenamos el cociente final y todos los residuos de cada división desde la última hasta la primera, previamente habiendo transformado cualquier residuo mayor a 9 en su equivalente de la tabla antes mencionada.

Nota: Puedes crear o aplicar otro método o algoritmo.

Escribe un programa que permita al usuario ingresar un número decimal, y el programa debe convertir el número a Hexadecimal, y al final debe imprimir el nuevo número.

Para esto tu programa debe implementar y usar la siguiente función:

- `getHexadecimal` recibe como parámetro `número en base decimal` y retorna `el número equivalente en base hexadecimal`.

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número decimal: 15963
El equivalente en Hexadecimal es: 3E5B
```

```
Ingrese un número decimal: 987654321
El equivalente en Hexadecimal es: 3ADE68B1
```

29. (Nivel 5) Los números perfectos son aquellos que son iguales a la suma de todos sus divisores exceptuando el mismo número, por ejemplo el número 6 es un número perfecto porque $6 = 1 + 2 + 3$. Por otro lado, un número primo es aquel que sólo es divisible por el 1 y por el mismo número, por ejemplo el número 7.

    Escribe un programa en python que reciba 2 números *min* y *max*, a partir de estos números tu programa deberá mostrar todos los números entre *min* y *max* que son perfectos o primos. Para esto tu programa debe implementar y usar las siguientes funciones:

    - `es_perfecto` recibe como parámetro un número entero positivo y retorna `True` si el número es perfecto o `False` en caso contrario.
    - `es_primo` recibe como parámetro un número positivo y retorna `True` si el número es primo o `False` en caso contrario.

Adicionalmente, si no hay ningún número perfecto o primo en el rango de *min* y *max*, deberá mostrar un mensaje apropiado. Algunos ejemplos de este programa serían:

```
Ingrese min: 1
Ingrese max: 7
2 3 5 6
```

```
Ingrese min: 8
Ingrese max: 10
No hay numeros perfectos ni primos
```