Programa:	Ingeniería en Computación
Curso:	Aseguramiento de la calidad del software
Asignación 3a:	Desarrollar programas alrededor de un dominio de problema y requerimientos funcionales abreviados
Instructores:	Ignacio Trejos Zelaya

Fecha límite de envío (tecDigital y por correo)**: viernes 2020.06.05, 23:55**
Valor: 5 puntos de la calificación del curso

1. Dominio: el calendario gregoriano

Investigar en la Web o en otras fuentes (enciclopedias, etc.) lo siguiente:

- Qué es el **calendario juliano.**
- Qué es el **calendario gregoriano.**
- Cuándo fue instituido el calendario gregoriano.
- Cuáles deficiencias o limitaciones del calendario juliano viene a subsanar el calendario gregoriano.
- Determinar la relación entre las fechas en que murieron los escritores Miguel de Cervantes y William Shakespeare.

2. Requerimientos funcionales generales

Con las propiedades del calendario **gregoriano** que Ud. investigó y el supuesto de que las fechas por tratar siempre están en una fecha **igual o posterior** a su entrada en vigencia en Roma, transforme los siguientes requerimientos generales en requerimientos funcionales más precisos. Todos los requerimientos tienen como contexto el dominio de las fechas en el calendario gregoriano. La programación debe desarrollarse en el lenguaje Python sin usar bibliotecas **ni** usar código ajeno a su equipo de trabajo (salvo lo indicado explícitamente abajo).

- R0 (fecha_es_tupla): Todas las *fechas* serán creadas como tuplas de tres números enteros positivos (**ternas**), en este orden: **(año, mes, día)**. *El resultado debe ser un valor booleano, True* o *False*.
- R1 (bisiesto): Dado un *año* perteneciente al rango permitido, determinar si este es bisiesto. *El resultado debe ser un valor booleano, True* o *False*.
- R2 (fecha_es_valida): Dada una *fecha*, determinar si ésta es válida según el calendario gregoriano. *El resultado debe ser un valor booleano, True* o *False*.
- R3 (dia_siguiente): Dada una *fecha válida*, determinar la fecha del día siguiente. *El resultado debe ser una fecha válida (tupla de tres números enteros positivos, que corresponde a una fecha en el calendario gregoriano, conforme a nuestra convención)*.
- R4 (dias_desde_primero_enero): Dada una *fecha válida*, determinar el número de días transcurridos desde el primero de enero de su año. Note que, dentro de un mismo año, el número de días transcurridos entre el primero de enero y el primero de enero, es 0. *El resultado debe ser un número entero.*
- R5 (fecha_hoy): Obtener la fecha 'del sistema' a partir de la invocación de la función de biblioteca de Pyhton `today()`[1],luego convertirla a una fecha válida – tupla (año, mes, día) – en el calendario gregoriano. *El resultado debe ser una fecha válida*
- R6 (edad_hoy): Dada una *fecha* válida, determinar la *edad* de la persona en años, meses y días cumplidos la fecha de *hoy. El resultado debe ser una tupla (año, mes, día)*; note que – en este caso – eso no es una *fecha válida*, sino una tupla con los tres componentes requeridos. *El resultado debe ser una tupla de tres números enteros no negativos.*

3. Restricciones no funcionales y técnicas

- Diseñe y programe todas las funciones que implementen los requerimientos funcionales del apartado anterior. **No** es válido reutilizar funciones o métodos de bibliotecas, salvo `date.today()`. Toda programación por entregar debe ser producto del trabajo de *su* grupo. Todo cálculo y lógica del programa debe ser *explicado*, sea en comentarios o en documentación externa al programa.

[1] E.g. `from datetime import date ; date.today()`. Ver https://docs.python.org/3/library/datetime.html

- Ud. tiene libertad para extender los requerimientos en cuanto a la detección de casos de error y la manera en que estos serán señalados vía programación.
- No nos interesa construir ni evaluar una interfaz de usuario amigable (por ahora).
- Satisfaga todos los requerimientos mediante programación en el lenguaje de programación Python (versión 3.8 o superior).
- Las funciones que implementan los requerimientos funcionales deben llamarse *exactamente* como se especifica en el segundo apartado (entre paréntesis, después del número que identifica al requerimientos), de lo contrario perderá los puntos correspondientes.
- Construya su programa de manera que el código sea legible.
- Debe trabajar en equipos de entre dos (2) y tres (3) miembros. Excepcionalmente, el profesor podrá autorizar el trabajo de manera individual, previa solicitud expresa por el estudiante interesado.
- Su solución debe ser entregada así:
  - Una **carpeta comprimida**, en formato **.zip**, que comprenda: portada (que identifique a los miembros del grupo), requerimientos funcionales, resumen de hallazgos respecto del calendario gregoriano, decisiones de diseño tomadas, código fuente de su solución, evidencias de las pruebas realizadas, análisis de resultados obtenidos. El código fuente debe estar en una sub-carpeta – dentro de la carpeta comprimida.
  - Todo su código fuente debe aparecer en un único archivo .py.

4. Entrega

- **Fecha de entrega: 2020.06.05, 23:59**
- El **archivo** con su solución debe llamarse "Asignación 3a - ", concatenado con los números de carnet de los estudiantes, ordenados *ascendentemente* (de izquierda a derecha) y separados por un guion medio rodeado por espacios (' - ').
- **Subir el material a la carpeta 'Asignación 3a' en el tecDigital.**
- Como respaldo, también debe enviar el archivo con su solución adjunta a un **mensaje** dirigido al profesor Ignacio Trejos Zelaya, itrejos@itcr.ac.cr, con copia a nuestra asistente, Ing. Verónica Mora Lezcano: vroimora@gmail.com. El asunto de sus mensajes de correo relacionados con este curso *debe tener siempre el prefijo*, "**IC-6831:** ". Ese prefijo debe ser seguido por "Asignación 3a - ", concatenado con los números de carnet de los estudiantes, ordenados ascendentemente (de izquierda a derecha) y separados por un guion medio rodeado por espacios (' - ').
- Esta asignación vale 5 puntos.