**PLANTEAMIENTO DE PROBLEMAS**

**Programación orientada a objetos**

1. Los exámenes de sangre son un procedimiento con el que se mide el estado de diferentes sistemas del cuerpo humano de forma indirecta. Esto debido a que el sistema circulatorio actúa como un medio de transporte para los diferentes compuestos químicos que son requeridos de con alguna funcionalidad en específica en el cuerpo humano. Permitiendo entonces que, por medio de la medición de estos compuestos químicos, y de la relación con su funcionalidad, sea posible diagnosticar el estado del cuerpo humano.

Se pide entonces implementar un sistema para la gestión de 5 diferentes tipos de exámenes de sangre, donde estos son: recuento sanguíneo completo, panel tiroideo completo, panel metabólico integral o perfil bioquímico completo, perfil o panel lipídico y ácido úrico.

**Funcionalidades del sistema:**

* Debe almacenar los exámenes según el paciente a quien se le realiza, permitiendo establecer un historial de todos los exámenes que se ha realizado el paciente.

La generación del historial del paciente y el almacenamiento de la información de este debe suceder dentro del sistema.

* La consulta de algún examen en específico se debe realizar por medio del ID específico de un examen, si se desea la realizar la consulta de todos los exámenes asociados a un paciente se deber realizar por medio del nombre de identidad de este.

**Simplificaciones:**

* Para simplificar el acercamiento a la generación de los diferentes exámenes de sangre, la diferencia que radicará en cada uno de estos es el tipo de compuesto químico que se mide.
* La información en general que determina un examen de sangre en general es: ID del examen, fecha de realización, método en que se realiza, nombre del médico que pide el examen y compuesto químico medido. La información asociada a la persona que se realiza el examen puede omitirse, debido a que la información de este se encuentra en el sistema, y dentro de esta información están los diferentes exámenes realizados.

1. Dentro del campo de conocimiento de la computación biomédica se encuentra el apoyo a actividades dentro de la biología. Dentro de esta, el campo de la genética se ha visto ampliamente beneficiada del área de la computación, permitiendo realizar análisis diversos con las diferentes secuencias de ADN. Existen algunas funcionalidades que se implementan reiterativamente sobre estas secuencias, por lo que para aumentar el rendimiento en que se ejecutan estas tareas implementar una generalización de estas es pertinente.

En este caso se pide implementar una generalización de una secuencia de ADN por medio de la programación orientada a objetos. Esta generalización debe permitir que se implementan las siguientes funcionalidades:

* Decir si la secuencia suministrada contiene bases nitrogenadas correctas.
* Encontrar la secuencia complementaria.
* Cargar un archivo .txt donde está almacenada la secuencia de ADN a analizar.
* Contar el número de bases nitrogenadas o elementos de la secuencia.

**Consideraciones**

Como información para resolver el ejercicio se menciona que la forma más simple de determinar una secuencia de ADN es por medio de un conjunto de letras como se muestra a continuación: AJUEYRNVKSOSKAMJFKALSDKFJ. La idea es entonces generar archivos .txt donde estén estas secuencias para luego ser leídas en la implementación con Python. Se debe tener en cuenta, como simplificación, que las únicas letras relevantes para una secuencia de ADN son: A, T, C y G; por lo tanto, si una secuencia ingresada contiene una letra por fuera de este grupo, esta no será válida como una secuencia de ADN.

Como comentarios finales, las letras A, T, C y G representan bases nitrogenadas, las cuales son: Adenina, Timina, Citocina y Guanina, respectivamente. Y estás se asociación con su secuencia complementaria de la siguiente forma:

Adenina 🡨🡪 Timina

Guanina 🡨🡪 Citocina

1. Implementar un sistema que se encargue del almacenamiento y carga de historias clínicas. Estás serán simuladas como archivos de texto .txt que se cargarán cuando el sistema sea encendido. El sistema entonces constará de un menú que en un inicio tendrá tres opciones: (1) Ingresar historia clínica, (2) Observar historia clínica y (3) Salir del sistema. La consulta de historias clínicas se puede realizar ingresando el id asociado a la historia clínica.

Para simplificación de la historia clínica los campos que esta contiene se sugieren que sean: Nombre del paciente, tipo de documento, número de identificación, id de la historia clínica (Este se debe generar automáticamente en el sistema, pero debe quedar registrada dentro del objeto asociado) y grupo sanguíneo. La consulta de la historia clínica se debe realizar con el id, no con el documento de identidad del paciente.

1. Implementar una clase que permita visualizar en el navegador el calendario ofrecido por la clase HTMLCalendar que ofrece el módulo *calendar* que viene por defecto en instalado en Python. Esta clase debe implementar dos funciones que abrirán el navegador automáticamente y mostrarán los calendarios pedidos. La primera función debe permitir observar el mes actual, o el mes que se le especifique del año actual. La segunda función debe permitir observar el calendario completo de un año ingresado, si no se especifica el año debe mostrar el calendario del año actual.
2. Durante el desarrollo de este ejercicio se implementarán diferentes funcionalidades pertenecientes al paquete de programación ***Numpy***. La intención principal de este es mostrar el uso de los vectores booleanos y sus posibles implementaciones cómo máscaras para la selección de determinados elementos en un array. Para esto se trabajará con una matriz que almacena cierta información y luego se deberán extraer ciertos vectores de esta dependiendo de lo que se pida.
3. Para este ejercicio se le pide que se realicen ciertas operaciones sobre una imagen que se carga, la imagen queda a disposición del estudiante. La idea es entonces usar las funcionalidades de Numpy para realizar las siguientes operaciones sobre la imagen:
   1. Dividir la imagen en 16 cuadrantes de igual dimensión.
   2. Asignar la siguiente distribución de colores a cada cuadrante.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | No modificar | No modificar |  |
|  | No modificar | No modificar |  |
|  |  |  |  |

Tenga en cuenta que cuando se lee una imagen esta se almacena como una matriz de 3x3, donde las dimensiones *x* e *y* representan la cantidad de pixeles de esta y la *z* los colores RGB.

Horiz Px

Vert Px

1. Integrar elementos de visualización e interfaces gráficas se suele realizar por medio de la integración de los paquetes de *PyQT* y *Matpotlib.* Sin embargo, existe un paquete que tiene ciertas funcionalidades de base que hace relevante el conocerlo. Este paquete se llama *PyQtGraph*, y durante el desarrollo de este ejercicio se observará cómo integrar diferentes elementos que este ofrece, a QtDesigner, observando también cómo acceder a la funcionalidad de estos elementos.

Para verificar que los conceptos mencionados quedaron claros se pide implementar una interfaz gráfica que permita al usuario ingresar un número y al oprimir un botón este se grafique, a medida que se vayan agregando más números se debe ir generando una curva en el objeto donde se están graficando los números. Se debe verificar que el usuario sí ingrese un valor numérico, en caso de no hacerlo debe aparecer una ventana emergente que alerte al usuario sobre la equivocación.

La interfaz gráfica puede tener la siguiente forma:

Graphical user interface

Description automatically generated with low confidence

1. Implementar una interfaz gráfica que permita la carga y visualización de imágenes. Esta debe tener la opción aplicar algún filtro sobre la imagen cargada.

El diseño de la interfaz puede ser como el que se muestra a continuación:

Graphical user interface, application

Description automatically generated

**Consideraciones:**

* Los filtros que se vayan a aplicar sobre la imagen se pueden implementar del módulo *scipy.ndimage.*
* El botón “Filtros” solo se puede usar cuando se ha cargado una imagen.
* La ventana que se abre cuando se oprime el botón de “Filtros” no bloquea la funcionalidad de la ventana principal.