A partir de la numeración, se decide evaluar el trabajo por partes, según cada archivo (.h/.cpp) que subió el ayudante. Misma diferencia encontrada en distintas partes del código serán omitidas. Todas las diferencias con el código del ayudante comienzan diciendo como esta en nuestro código

Notamos que los archivos se llaman de formas distintas, y nosotros hicimos un header global donde guardamos las librerías, el enum, etc. Todo esto lo consideramos alternativas.
Respecto al UML:

- No sabíamos que los enum deben ir en el UML (error nuestro).
- No realizamos una relación entre cPaciente y cLaboratorio, ya que la relación del laboratorio con el paciente se da a través del centro de testeos. Desde el punto de vista del ayudante, entendemos que los pacientes dependen de los laboratorios, y por eso en nuestro trabajo establecimos una composición del paciente al centro, pero no consideramos que si sin el laboratorio, el paciente deja de tener sentido. Consideramos que lo nuestro está mejor pensado.
- Uso de agregación del laboratorio hacia el centro. Tiene una relación fuerte, pero pensamos al laboratorio como algo funcional sin el centro. Sin embargo, el ayudante plantea que sin centro, el laboratorio deja de tener sentido. Pensamos que tiene más sentido la justificación del ayudante, error nuestro por ende.
- Nuestro programa fue pensado para que de base se envíen pacientes y se reciban, y a su vez se mandan al laboratorio. El programa del ayudante también plantea sets de centros, laboratorios y pacientes al principio del programa, pero mediante el UML sugiere que puede haber 0 pacientes en el centro, o en el laboratorio, o en la clase paciente, pero para eso se debe modificar el programa, entonces creemos que no se debería tener en cuenta el caso donde no hay pacientes, o/y centros, o/y laboratorios.

1. Enumns.h:
    - No utilizamos <summary> para comentar. Consideramos que es un pequeño error, ya que comentar sin summary puede llegar a cumplir el mismo objetivo que con él, pero en determinadas situaciones puede ser muy útil.
    - No pasamos el enum a string. Nosotros proponemos la alternativa de imprimir el número del resultado, aclarando antes a que refiere cada número. Tuvimos en cuenta que la impresión en pantalla no es para el paciente.
2. Enums.cpp (lo escrito en el segundo ítem del punto anterior)
3. cPaciente.h:
    - No uso de pragma regions. Creemos que para la organización del código esto es un error nuestro. Fue debido a que algunos del grupo desconocían esta utilidad, y en cierta medida por falta de tiempo.
    - Modificador de acceso público y después privado. Nuestra alternativa propone primero el modificador público, y después el privado, dado que es lo que se frecuenta.
    - Los atributos que conforman los síntomas fueron enteros. Es verdad que usar booleanos le da una sensación más realista a los síntomas, pero nuestra alternativa propone que sean enteros (0 falso , 1 verdadero) para poder sumar luego los síntomas sin tener que hacer un algoritmo extenso
    - Definición de la función getResultadoTesteo() en el Header. Creemos que esto es una mejora por nuestra parte, ya que las funciones cuya definición es “corta”, es óptimo hacerlo en el header
    - No implementación de GetDNI. Nuestro programa ofrece una alternativa funcional sin tener en cuenta el DNI del paciente.
4. cCentrosTesteo.h:

- Nuestra declaración del número del ID centro, es string. Dado que no cambia su valor a lo largo del programa, consideramos que esto es una pequeña mejora por nuestra parte en el aspecto del programa.
- No utilizamos unsigned para aclarar que la variable sólo tomará valores positivos. Pequeño error por nuestra parte.
- Llamar a los atributos distinto que a los parámetros de los métodos. Según lo visto en las clases prácticas, lo ideal es cambiarle el nombre para no generar confusión. Esto lo consideramos una mejora por nuestra parte.
- Nuestro constructor no usa parámetros por defecto. Esta alternativa se da principalmente porque en el main nosotros directamente le asignamos los parámetros a cada constructor de cada centro.
- En el método baja paciente, enviamos como parámetro el resultado de su testeo. Error por nuestra parte, no tuvimos en cuenta que a través de los atributos de los pacientes podíamos acceder al resultado de su testeo, y de esa forma compactar mejor el programa.

5. cCentros_De_Testeo.cpp:
    - Nuestro constructor hace dinámico a los punteros paciente por el hecho de que, según lo visto en clase, se debe hacer dinámica la clase débil dentro de una composición. Según el UML presentado en GitHub por parte del ayudante, el establece 3 composiciones y no hace dinámico ningún puntero sobre las clases débiles. Por otra parte, si existe new en el constructor, existe delete en el destructor (no existen los new en el tp del ayudante, por ende tampoco existen los delete).
    - Dentro del método mandar testeo, no chequeamos que esté asignado él laboratorio al centro, y el verificar que el laboratorio esté lleno lo hacemos dentro del método recibir muestra. Lo primero lo consideramos error, lo segundo alternativa.
    - En el método baja paciente, debido a la no especificación de la consigna, no imprimimos a quién se dio de baja. Lo consideramos una alternativa.
    - En el método asociar laboratorio, no chequeamos que se intente asociar un laboratorio más de dos veces. Error por nuestra parte
    - El método to_string lo hicimos mediante la clase stringstream. Es lo mismo, pero hecho de otra forma. Lo consideramos una alternativa.
6. cLaboratorio.cpp:
    - En el método recibir muestra, no tenemos en cuenta que el paciente 1 no tenga el mismo DNI que el segundo. Por otra parte,analizamos la muestra cuando se consigue ingresar la muestra directamente, para así ahorrar todo el procedimiento de volver a chequear que no sea NULL el paciente. Lo primero lo consideramos error, y lo segundo una mejora (en eficiencia del programa).
    - En análisis de muestra, el resultado lo seteamos en la clase paciente luego de avisarle (alternativa). Luego, generamos la variable resul porque no sabíamos que se podía utilizar el operador de resolución de alcance (::) para los enum. No verifico que exista el paciente porque este método lo utilizo apenas asigno al paciente (mejora). No volvimos a null al paciente luego de avisarle (error grande)**.**
7. TP1_2022.cpp:
    - Declaración de arrays para objetos de una misma clase. Podríamos considerarlo como una gran alternativa, que modificara el funcionamiento del programa debido a que nosotros fuimos implementando los métodos mediante bucles.
    - La impresión en pantalla la hicimos al final del programa directamente (alternativa).
    - No se llama al análisis de muestra porque se hace dentro de mandar testeo. Consideramos esto una optimización (mejora).
    - La baja del paciente y la desasociación del laboratorio la hacemos en el main. El ayudante nunca llama a alguno de los dos métodos que definió, un gran error por su parte.