Producto de la situación sanitaria actual, se hace imperativo el realizar un seguimiento informático al proceso de vacunación con un control exhaustivo de pacientes que han logrado ser vacunados e identificar aquellos que aún están pendientes de vacuna.

Se dispone de una base de datos de ciudadanos ficticios, consistente en un archivo CSV que contiene apellidos, nombres, RUN, fecha de nacimiento, género y que corresponderá a personas (pacientes) elegibles a recibir una vacuna. Puede obtener una copia de este archivo desde la siguiente URL: https://bit.ly/3uwuQmg

El vacunatorio responsable de dar curso al proceso de vacunación, dispone de 3 variantes comerciales de vacunas para inoculación en primera dosis hoy:

A. **Sinovac**: recomendable para personas de entre 18 y 80 años, con un stock disponible de 90 dosis en total.
B. **Pfizer**: recomendable para personas entre 15 y 55 años, con un stock de 120 dosis en total.
C. **AstraZeneca**: recomendable para mujeres mayores de 41 años y hombres de 18 o más años, con un stock disponible de 70 dosis en total.

Para dar soporte a este proceso, se le pide a Ud. que desarrolle un programa en C++ considerando los siguientes lineamientos:

- Modele el stock de dosis existentes mediante la construcción de una única clase, **Dosis**, que tendrá atributos (propiedades) tales como (i) **tipo o variante comercial** de la vacuna, (ii) **edad mínima** para inoculación (inclusive), (iii) **edad máxima** para inoculación (inclusive), el (iii) **número identificador** de serie y (iv) si está utilizada o bien disponible. Al respecto, cada dosis de una vacuna en particular corresponderá a una instancia de la clase **Dosis**. Un objeto (instancia) de Dosis sólo podrá ser utilizada una única vez.

- Modele el universo de pacientes elegibles para recibir una dosis mediante la construcción de la clase **`Paciente`**, que debe considerar como atributos la (i) **fecha de nacimiento** del paciente, (ii) el **RUN**, (iii) nombres y apellidos. Para las propiedades fecha de nacimiento y RUN se deberá considerar las clases **`RUN`** y **`Fecha`** descritas más adelante.

  `Paciente` debe tener un método `int edad()` que calcula y devuelve los años cumplidos del paciente.

- La clase **`RUN`** representa el RUN de un ciudadano. Su único atributo que posee representa el valor de un RUN como un string que no contiene puntos, con guión y con dígito verificador en mayúsculas después del guión. Ejemplo "12345678-K".
  El constructor de **`RUN`** inicializa su propiedad RUN con valor nulo. La clase **`RUN`** debe internamente disponer de un mecanismo de validación del dígito verificador cada vez que se intente asignar un nuevo valor a su propiedad (cambio de estado). **El cambio de estado de la instancia RUN solo debe ocurrir si el valor RUN que se está asignando es válido**[1], retornando un valor `true` como resultado de la acción. Si el valor de RUN no es válido, entonces no hay cambio de estado y la acción retorna un `false`.

- La clase **`Fecha`** representa una fecha mediante el uso de 3 propiedades de tipo entero que permiten representar el año, mes y día. El constructor de **`Fecha`** inicializa la fecha al 01 de enero de 1900. Cada vez que se intente asignar un nuevo valor de fecha ésta deberá ser previamente validada. **Solo se podrá modificar el estado de la instancia cuando la fecha a asignar sea válida** (ejemplo para 25/05/2021) retornando un valor `true` como resultado de la acción. En el caso de que la fecha sea incorrecta (ejemplo 30/02/2021), se deberá retornar un `false`.

- Cada dosis que se aplique a un paciente deberá estar representada por una instancia de la clase **`Inoculación`**, que tendrá solo 3 propiedades: una instancia **`Paciente`**, otra instancia de **`Dosis`** aplicada, y la fecha en que la dosis ha sido aplicada (suponga la fecha actual). Estas instancias de **`Inoculación`** deben estar organizadas como una Lista Lineal Simple (LLS 1).

Su programa principal debe ofrecer un menú con **exactamente** las siguientes 8 opciones:

(a) **Informar fecha actual**: imprime la fecha del sistema en formato dd/mm/aaaa
(b) **Cargar base de datos de pacientes**: accede al contenido de un archivo CSV que deberá llamarse *pacientes.csv*, traspasando su contenido a memoria del computador en una LLS 2, siendo cada nodo de la lista una instancia de tipo **`Paciente`**.

[1] Investigar en Internet algoritmo de cálculo de dígito verificador RUN/RUT chileno

IMPORTANTE: solo se incorpora un nuevo nodo a la LLS cuando el RUN y la fecha de nacimiento sean válidos. En caso contrario el registro se debe descartar. Al finalizar el proceso se debe informar en pantalla cuántos pacientes (registros) han sido leídos del archivo y cuántos han sido cargados correctamente en memoria.

(c) **Resultado de carga de datos de pacientes**: informa en pantalla cuántos pacientes han sido leídos desde el archivo y cuántos han sido cargados correctamente.

(d) **Crear dosis**: con base en las existencias en el vacunatorio y representándolas mediante una LLS 3 en que cada nodo es una instancia de la clase **Dosis**.

(e) **Iniciar vacunación**: inicia un proceso de vacunación mediante la invocación a la función externa `IniciarVacunacion(parámetros)` la que deberá considerar lo siguiente:

(i) Obtener el siguiente paciente a inocular desde la lista de pacientes, determinando el tipo de vacuna recomendada según sus antecedentes.

(ii) Inocular a todos los pacientes (que se pueda), asignándole a cada uno la primera dosis recomendable disponible y marcando dicha dosis como utilizada. Debe agregar cada inoculación a la lista de objetos **`Inoculación`**.

(iii) Informar al final del proceso:

- Cantidad de dosis utilizadas por tipo de vacuna.
- Cantidad de dosis disponibles por tipo de vacuna.
- Cantidad de pacientes inoculados según los rangos etarios siguientes:
  - 0 a 18 años
  - 19 a 25 años
  - 25 a 35 años
  - 45 a 65 años
  - +65 años

(f) **Consultar paciente**: solicita por consola el RUN del paciente y arroja por pantalla sus datos personales y con qué dosis fue inoculado (**número identificador** y **marca**). En caso de que el paciente no haya sido inoculado se debe informar esto por pantalla. Si el RUN fue mal ingresado por consola se debe informar al usuario.

(g) **Pacientes no vacunados**: Listado completo por consola que indica el RUN, nombre, edad y género de los pacientes que no recibieron dosis.

(h) **Salir**: finaliza la ejecución del programa.

Cada opción del menú debe llamar a una función externa que será la responsable de ejecutar la lógica de programa necesaria para la funcionalidad solicitada.

Video: explicando su código y mostrando su programa funcionando

Pauta de evaluación:

Funcionalidad	**Comentarios/indicaciones**	**Puntos**
El código está organizado con archivos .h y .cpp separados por cada clase	Cada clase debe tener su archivo header y el código fuente por separado	10
Correcta implementación de clase `RUN`	Debe contar con constructor, destructor, *setters*, *getters* y ver. Implementa constructor que inicializa sus atributos con valores por defecto. Posee método validador privado y setter que llama al validador (devuelve true si los parámetros están correctos y realizan el cambio, false si los parámetros no pasan la validación y no realizan el cambio)	10
Correcta implementación de clase `Fecha`	Debe contar con constructor, destructor, *setters*, *getters* y ver. Implementa constructor que inicializa sus atributos con valores por defecto. Posee método validador privado y setter que llama al validador (devuelve true si los parámetros están correctos y realizan el cambio, false si los parámetros no pasan la validación y no realizan el cambio)	10
Correcta implementación de clase `Dosis`	La clase debe tener atributos privados. Debe contar con constructor(es), destructor, *setters*, *getters* y ver.	10

Correcta implementación de clase `Inoculación`	La clase debe tener atributos privados. Debe contar con constructor(es), destructor, *setters*, *getters* y ver.	10
Correcta implementación de clase `Paciente` (composición de clases `RUN` y `Fecha`)	La clase es compuesta. Debe contar con constructor(es), destructor, *setters*, *getters*, ver y edad.	10
Captura fecha del sistema operativo	Para obtener el puntaje de este ítem debe investigar y lograr capturar la fecha actual desde el sistema operativo. Si no lo logra, puede pedir la fecha actual por teclado (pero no obtendrá puntos por esto)	10
Lectura y carga de la base de datos de pacientes.	Lee el archivo y carga los elementos correctos en la lista, omitiendo aquellas entradas que no sean válidas.	20
Crear las dosis de las vacunas	Vea recomendaciones al final de este documento.	10
Realizar el proceso de vacunación	Simula la vacunación de las personas en el mismo orden de aparición dentro del archivo de pacientes. Se asigna la primera vacuna (dosis) que le sirva y se encuentre disponible según edad y género del paciente.	20
Obtener y mostrar estadísticas finales	Muestra las estadísticas finales según rango etario.	20
Mostrar los datos personales y con qué dosis (**número identificador**) se vacunó	Dado un RUN, busca a la persona en la lista y luego en base al número de serie busca la vacuna en la otra lista y muestra los datos.	20
El listado de personas que no se vacunaron	Obtiene el listado de todas las personas que no se vacunaron.	20
Video explicativo	Crea un video explicando su diseño. Tiempo: 3 a 5 minutos de exposición. No se requiere edición especial de video, pero sí se debe escuchar su voz y ver su rostro. Coloque el link del video en Youtube como comentario al inicio de su código.	20
	TOTAL PUNTOS	**200**

Observaciones:

- Fecha de publicación: **lunes 31 de mayo de 2021, 18:00 hrs.**
- Fecha de entrega: **viernes 25 de junio 2021, 18:00 horas**
- Nota: Tome las precauciones del caso para entregar la tarea dentro del plazo. No se aceptan tareas fuera de plazo.
- El trabajo es desarrollado **en forma INDIVIDUAL**, colocando la identificación de la persona, sección y profesor(a) como comentario dentro del código.
- Si el proyecto es copia, obtendrá nota mínima 1.0

Recomendaciones:

- Con respecto a crear las dosis de las vacunas se recomienda implementar la siguiente solución:

```
numero_de_serie = 10000;
for desde 1 hasta CUANTOS_SEAN
{
	numero_de_serie = numero_de_serie + 3;
	dosis nueva con datos de sinovac, con numero_de_serie
	agregar dosis nueva a la lista
}
for desde 1 hasta CUANTOS_SEAN
{
	numero_de_serie = numero_de_serie + 7;
	dosis nueva con datos de pfizer, con numero_de_serie
	agregar dosis nueva a la lista
}
for desde 1 hasta CUANTOS_SEAN
{
	numero_de_serie = numero_de_serie + 5;
	dosis nueva con datos de aztrazeneca, con numero_de_serie
	agregar dosis nueva a la lista
}
```

- **Por simplificación, suponga que las personas solo requieren una primera y única dosis de la vacuna**.