```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "$ Examen practico" << endl;
    cout << "$ 1er Parcial" << endl;
    cout << "-----------------" << endl;
    cout << "Que comience el juego Uwu" << endl;

    return 0;
}
```

Aspectos a evaluar

- Que funcione (._.)
- Codigo limpio y ordenado.
- Que el tipo de dato de los atributos y los metodos sea el mas apropiado.

Tomar en cuenta

- Copia de internet: **cancelacion automatica**.
- Codigo duplicado: **cancelacion automatica para ambos**.

Instrucciones

Implementa la clase segun el numero de examen que te corresponde. Se deben implementar por lo menos 3 objetos (instancias) de la clase en el `main.cpp` para realizar pruebas.

Examen 1

Crear una clase llamada `Cuenta` que tiene los siguientes atributos (con los respectivos getters y setters):

- **titular**
- **saldo**

Debe tener los siguientes metodos:

- **ingresarSaldo(`cantidad`)** *aumenta el saldo en cierta cantidad.*
- **retirarSaldo(`cantidad`)** *reduce el saldo en cierta cantidad.*

Para el metodo `retirarSaldo` si la cantidad a retirar es mayor a el saldo disponible debe retornar `0`

Examen 2

Crear una clase llamada `Persona` que implemente los siguientes atributos, con sus respectivos getters y setters:

- **edad**
- **sexo** (`h` para hombre y `m` para mujer)
- **ine**
- **peso**
- **altura**

Los metodos que debe implementar la clase son:

- **calcularMasaCorporal()** *Calcula el IMC con base a los atributos de la instancia de la clase, si el indice indica por debajo del peso normal, retorna -1, si el IMC es normal retorna 0, si es sobrepeso retorna 1*
- **esMayorDeEdad()** *Retorna verdadero o falso de acuerdo al valor de edad del atributo edad.*

Examen 3

Implemente una clase llamada `Curp`, que cuente con los siguientes atributos, con sus respectivos getters y setters:

- **nombre**
- **apellido_paterno**
- **apellido_materno**
- **nacionalidad**
- **entidad_de_nacimiento**
- **nacimiento_dia**
- **nacimiento_mes**
- **nacimiento_anio**
- **genero**

La clase tendra los siguientes metodos:

- **generar()** *Retorna la generacion de la curp con base a los atributos de la clase.*

Examen 4

Crea una clase llamada `Password` con los siguientes atributos, con sus respectivos getters y setters:

- **cadena**
- **longitud**

La clase debe implementar los siguientes metodos:

- **esFuerte()** *Retorna `verdadero` si la contrasenia, (que se contiene en el atributo cadena) es fuerte o `falso` si no, es fuerte cuando: contiene al menos una mayuscula, un numero y un simbolo.*
- **generar()** *Genera una cadena aleatoria de longitud dada por el atributo `Longitud`, y la almacena en el atributo `cadena`*

Examen 5

Crear una clase llamada `CuerpoEnMovimiento`, que implemente los siguientes atributos, con sus respectivos getters y setters:

- **masa**

- **posicion_inicial**
- **posicion_final**
- **tiempo_del_recorrido**

Debe implementar los siguientes metodos:

- **calcularVelocidad()** *Retorna el calculo de la velocidad del cuerpo en movimiento.*
- **calcularEnergiaSinetica()** *Retorna el calculo de la energia sinetica del cuerpo en movimiento.*

Examen 6

Crear una clase llamada `Dado`, con el siguiente atributo, con sus respectivos getters y setters:

- **lados**

E implementando el siguiente metodo:

- **lanzar()** *Retorna un numero aleatorio en tre 1 y el valor del atributo `Lados`.*

Examen 7

Crear una clase llamada `Estadistica`, que tenga el siguiente atributo:

- **calificaciones[10]** (*es decir, una lista de 10 calificaciones*)

Con los siguientes metodos:

- **agregarCalificacion(`calificacion`)** *Si aun hay espacio en la lista de 10 calificaciones, agrega la calificacion dada a la lista.*
- **calcularPromedio()** *Retorna el valor del promedio de las calificaciones.*
- **calcularMedia()** *Retorna el valor de la media de las calificaciones.*
- **calcularModa()** *Retorna el valor de la moda de las calificaciones dadas.*

Examen 8

Crea una clase llamada `Producto`, con los siguientes atributos, con sus respectivos getters y setters:

- **nombre**
- **caducidad_dia**
- **caducidad_mes**
- **caducidad_anio**

y los siguientes metodos:

- **diasParaCaducar(`dia_actual`, `mes_actual`, `anio_actual`)** *Retorna el numero de dias restantes para que el producto caduque, con base en en los atributos de caducidad y la fecha actual dada.*

Examen 9

Implementa una clase llamada `Pitagoras`, que cuenta con los siguientes atributos, con sus respectivos getters y setters:

- **base**
- **altura**

e implemente el siguiente metodo:

- **calcularHipotenusa()** *Retorna el calculo de la hipotenusa dados los atributos de la clase.*

Examen 10

Implementa una clase llamada `CalculadoraVectorial`, que implemente los siguientes atributos, con sus respectivos getters y setters:

- **vector1[2]** *(Lista de un par de numeros que representan al primer vector bidimensional.)*
- **vector2[2]** *(Lista de un par de numeros que representan al segundo vector bidimensional.)*

debe implementar los siguientes metodos:

- **calcularSuma()** *Retorna el resultado vectorial de la suma de los vectores 1 y 2.*
- **calcularResta()** *Retorna el resultado vectorial de la resta de los vectores 1 y 2.*
- **calcularProductoEscalar()** *Retorna el calculo del producto escalar entre ambos vectores.*
- **calcularDistanciaEscalar()** *Calcula la distancia escalar entre los 2 vectores.*

Examen 11

Crea una calse llamada `Impuestos`, que implemente los siguientes atributos:

- **total_ingresos**

y los siguientes metodos:

- **agregarIngreso(`cantidad`)** *Auomenta el ingreso total segun la cantidad dada.*
- **calcularISR()** *Retorna el calculo del impuesto a pagar por ISR.*
- **calcularIVA()** *Retorna el valor del impuesto por IVA a pagar.*
- **totalDeducciones()** *Retorna el total de deducciones que se aplican por los impuestos.*
- **ingresoNeto()** *Retorna el valor de los ingresos totales despues de impuestos.*