Ejercicios Programación Avanzada

Ejercicio 1

Considere la siguiente estructura de la clase Reloj:

```
class Reloj
 {
  private:
      int hora;
      int minutos;
      int segundos;

}
```

La clase cuenta con los correspondientes métodos *set* y *get*, y cuenta con un método adicional llamado **ajustarHoraOficial**.

Se pide:

Programe el método void **ajustarHoraOficial (Reloj *ho)** o bien **ajustarHoraOficial (Reloj ho)** (elija uno de ambos) que recibe como parámetro un objeto de la misma clase Reloj que **cuenta con la hora oficial**. El método debe ajustar la hora del objeto actual a la hora oficial, e indicar, **con un mensaje por pantalla**, si el reloj estaba: A LA HORA, ADELANTADO o ATRASADO.

Ejercicio 2

Un profesor se dedica sólo a realizar clases particulares a grupos de alumnos, las que realiza en su oficina particular. Su agenda diaria es estricta, la divide en 10 bloques diarios que operan ininterrumpidamente desde las 10 de la mañana.

Cuando lo contacta un grupo de estudio les solicita los siguientes datos: Número de alumnos, nombre y teléfono del alumno responsable.

El grupo puede solicitar un bloque en particular o bien lo escoge el profesor en forma automática.

Para el manejo de grupos el profesor utiliza una clase de C++ denominada **Grupo**, y para su trabajo diario una llamada **Agenda**, la agenda tiene los bloques del día para su asignación.

El profesor recauda por cada alumno $10.000 y si el grupo lo conforman más de 3 individuos tendrán un 20% de descuento.

En base a lo anterior se pide:

- Programe la clase Grupo completa, incluyendo un método que imprima los datos del grupo
- Programe la clase Agenda considerando los siguientes métodos:
    - bool agregarGrupoAutomatico (Grupo &gx) / bool agregarGrupoAutomatico (Grupo *gx), que recibe un grupo de estudio y retorna si se pudo hacer la asignación de algún bloque disponible para dicho grupo. **Obs utilice el método de acuerdo con lo visto con su profesor**

- bool agregarGrupoBloque (Grupo &gp, int bloque) / bool agregarGrupoBloque (Grupo *gp, int bloque), que recibe un grupo de estudio y retorna si fue posible asignar dicho grupo en el bloque solicitado. **Obs utilice el método de acuerdo con lo visto con su profesor**
- int RecaudacionDiaria ( ) que retorna el monto a recaudar según la información almacenada en la agenda

- Programe el método main de modo que active una agenda diaria y luego asigne un grupo en forma automática
- Programe la función main de modo que genere una agenda diaria y luego reciba por teclado los datos de un grupo, lo asigne en forma automática e imprima el monto total a recaudar.

´

Ejercicio 3

La comitiva olímpica de Chile viaja en un avión que permite llevar un máximo de 100 deportistas. Los deportistas tienen nombre y especialidad deportiva que nunca cambian y un sueldo que solo puede subir. En base al enunciado anterior, se pide:

a. Genere la clase **Deportista** que permita crear deportistas y posea los atributos y métodos get y set correspondientes.
b. Genere la clase **AvionOlimpico** que se compone de un arreglo de 100 deportistas. El avión comienza inicialmente vacío (sin deportistas) y debe especificar (no programar) los métodos correspondientes a esta clase.
c. Para la clase **AvionOlimpico** genere un método llamado **ingresarDeportista** que permita ingresar deportistas al Avión. Considere que los asientos se van llenando ordenadamente desde el asiento 1 al 100 y que un deportista sólo puede ocupar un asiento.
d. Para la clase **AvionOlimpico** genere un método llamado **imprimirComitiva** que imprima todos los datos de cada deportista que esté ingresado en el Avión apoyándose, obligatoriamente, en los métodos de la clase Deportista. La impresión debe ser de la forma:

Asiento 1: Juan Perez / Ciclismo / 500000
Asiento 2: José Pérez / Karate / 450000
…
…
Asiento N: Carla Soto / Gimnasia / 550000

Importante
Si un asiento no tiene ocupante, no debe imprimir información.

Ejercicio 4

Considere la clase Fracción que tiene dos atributos enteros, numerador y denominador, sus métodos set's y get's y un método que imprime la fracción (numerador "/" denomindor).

Programe en la clase Fracción el método simplificar(), que simplifica la fracción en caso de que sea posible hacerlo.

Programe el método sumar que recibe como parámetro dos objetos de tipo Fracción y devuelve un objeto Fracción con la suma de las dos fracciones. La Fracción que se devuelve debe estar simplificada.

Ejercicio 5

Una institución financiera ofrece depósitos a plazo en pesos (monto en dinero) cuyos plazos están en los **30, 60 o 90 días**. Cada plazo tiene un **interés** (porcentaje) que puede ser distinto por cada **cliente** (para identificar al cliente solo utilizaremos su nombre) y se decide al momento de crear el depósito. Al momento de finalizar el plazo del depósito, el cliente puede:

- ✓ Retirar el dinero, por lo que retira **capital** más **intereses.**
- ✓ Renovar el depósito por igual o distinto plazo.

Sin perjuicio de lo anterior, el cliente puede solicitar su dinero antes de cumplir el compromiso. En dicho escenario, se le castiga con los intereses (no se le pagan) y pierde el 1% del capital invertido por concepto de costos operacionales.

De acuerdo con el caso antes expuesto, se pide:

a) Genere la clase **Deposito** con los atributos y el constructor necesario para el enunciado. Considere que un depósito no puede comenzar sin un monto asociado.
b) Cree el método **retirarPlazoCumplido** que retorne el dinero correspondiente cuando finaliza el plazo de un depósito.
c) Construya el método **renovarDeposito** que permita renovar el depósito considerando el monto nuevo de dinero (capital + intereses) y un nuevo plazo asociado.
d) Edifique el método **retiroAnticipado** que permite retirar el dinero antes de cumplir el plazo y debe actualizar el depósito a cero y retornar el monto en dinero correspondiente a este evento.

Ejercicio 6

La descarga de RILES (Residuos industriales líquidos) de las plantas industriales puede provocar graves problemas al medio ambiente por lo que es necesario controlar el flujo que se descarga hacia redes de alcantarillado o incluso cursos de aguas naturales. Dicho control se hace “en puntos de muestreo - PM”, los cuales se identifican con un código numérico, un nombre, la cantidad de PH medido y la Temperatura registrada.

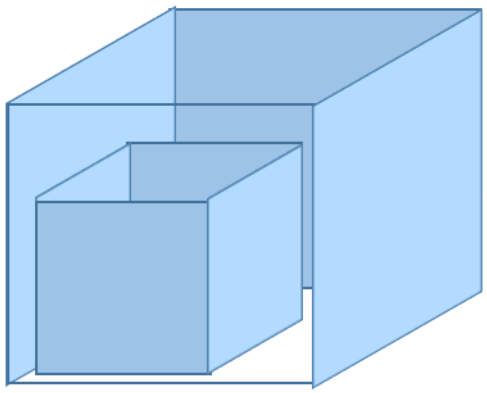
Así mismo, toda planta industrial posee 12 puntos de muestreo, inicialmente vacíos, y además posee un código numérico, un nombre y número de la región a la que pertenece.

En base a lo anterior y como una forma de apoyar la medición diaria que se hace en una planta industrial se pide:

a) Programe la clase **Punto de Muestro** con sus atributos, constructor y métodos get y set que correspondan.
b) Genere el método **imprimirPM** que imprima los datos asociados a un PM.
c) Programe la clase **Planta Industrial** con sus atributos, constructor y métodos get y set que correspondan (incluyendo el método agregar punto de muestreo).
d) Programe un main sencillo en el que cree 3 puntos de muestreo, una planta industrial y luego introduzca en la planta los 3 puntos de muestreo generados.

Ejercicio 7

Implemente una clase 'caja' que tenga dimensiones x,y,z. En dicha clase caja debe desarrollar un método llamado 'cabe' que recibe como parámetro otro objeto de tipo caja. El método debe retornar 'true' si es que la caja ingresada como parámetro cabe dentro de quien lo invoca y 'false' si no cabe. El método además debe imprimir por pantalla la diferencia de volumen entre las cajas.



Ejercicio 8

En base a la siguiente clase:

```
class Avion{
  private:
 int tripulantes;
  int velocidad;
  int combustible;
  int x;
  int y;
  public:
  Avion(int tripulantes,int velocidad, int combustible,int x ,int y){
    this->tripulantes = tripulantes;

    this->x=x;
    this->y=y;
  }
  float consumo(int distancia){
    return ((this->tripulantes*distancia)*this->velocidad);
  }
 };
```

Cree una clase PortaAviones que tenga un arreglo con capacidad máxima para 100 Aviones. Además debe tener coordenadas 'x' e 'y' cuyos valores son ingresados como parámetros del constructor.

La clase PortaAviones debe tener un método 'agregar' que agregue un nuevo avión al porta aviones. Los nuevos aviones tienen las mismas coordenadas que el porta aviones.

La clase PortaAviones debe tener un método "float atacar(int x, int y)". El método debe retornar el consumo total de combustible de toda la flota de aviones para recorrer la distancia hacia el punto especificado en los argumentos del método. Utilice el método consumo de la clase Avión (*float consumo(int distancia)*) para obtener los valores individuales de cada avión.

$$\text{distancia}=(x2\text{-}x1)2+(y2\text{-}y1)2$$

Ejercicio 9

En un mapa existe un máximo de 100 ciudades. Cada ciudad se encuentra representada por una coordenada de dos dimensiones (x,y), junto con un nombre.

1. Implemente la clase mapa con un arreglo de ciudades y un método para agregar ciudades al arreglo.
2. Implemente la clase Ciudad con los get y set correspondientes según los atributos.
3. La clase Mapa debe tener un método que imprima por pantalla el punto medio entre todas las ciudades, es decir el promedio de las coordenadas X y el promedio de las coordenadas Y de cada ciudad.

Ejercicio 10

La clase ConsumoVehiculo permite a una familia saber cuánto consume un automóvil, cuánto puede recorrer y el costo de recorrer una determinada cantidad de kilómetros.

La clase tiene los siguientes atributos: marcaVehículo, rendimiento (km/l), bencinaDisponible (litros), capacidadEstanque(litros).

En base a lo anterior se pide:

1. Cree la clase ConsumoVehiculo con constructor vacío, constructor con parámetros, métodos set's, get's e imprimir.
2. Programe el método cuantoPuedeRecorrer() que devuelve la cantidad de km que puede recorrer el vehículo en función del rendimiento y la bencina disponible
3. Programe el método int llenarEstanque(int precioLitro), que llena el estanque de bencina y devuelve el monto a pagar.
4. Programe el método bool validarDistancia(int kmarecorrer), que devuelve verdadero si puede recorrer los kmarecorrer y falso en caso contrario, todo en función de la bencinaDisponible.

Ejercicio 11

La clase MisNotas tiene las notas de una asignatura de un alumno, se compone de un arreglo de float de tamaño 10 (que es el máximo de notas que un alumno puede tener). El arreglo se inicializa con valores en 0 que implica ausencia de notas. En base a lo anterior se pide:

1. Cree el constructor que inicializa las notas en 0
2. Cree el método void agregarNota(float nota) que agrega una nota buscando la primera posición del arreglo en 0 y agrega la nota, si no hay espacio disponible se reemplaza la nota más baja, si todas las notas son superiores se envía mensaje que la nota no pudo ser agregada.
3. Cree el método float obtenerPromedio(), que devuelve el promedio de las notas disponibles (distintas de 0)

Ejercicio 12

Cree una Clase **NumeroComplejo**, que permita representar un número complejo (tiene parte entera y parte imaginaria), la clase debe tener sólo el constructor con parámetros, los métodos set's, get's y un método que imprima el número imaginario y lo devuelva de la forma a +bi, si los valores de a ó b son 0 no se deben mostrar, si ambos son 0 debe mostrar 0.

Cree la clase **SegundoGrado**, que permita registrar una ecuación de segundo grado, la clase tiene los parámetros a, b y c respondiendo a la fórmula $ax^2 + bx + c = 0$. La clase debe implementarla sólo con el constructor con parámetros y en forma adicional dos métodos obtenerRaiz1 y obtenerRaiz2, ambos métodos devuelven un objeto de tipo NumeroComplejo con la solución de la raíz correspondiente.