Apellidos, Nombre:_____ DNI: _____

INSTRUCCIONES

Realiza los ejercicios tal como se detallan en cada uno de ellos. Se proporcionan varios folios vacíos para poder anotar lo que se necesite de la resolución del examen, el contenido de esos folios no se evaluará, pero serán entregados al profesor.

Descargue el zip proporcionado en Moodle llamado "*AlternativaBloqueI-ApellidosNombre.zip*". Renombre la carpeta sustituyendo "*ApellidosNombre*" por sus dos apellidos y nombre. Por ejemplo, si mi nombre es Antonio Manuel Durán Rosal, el directorio pasa de "*AlternativaBloqueI-ApellidosNombre.zip*" a llamarse "*AlternativaBloqueI-DuranRosalAntonioManuel.zip*" (recuerde que no puede haber **ni espacios ni caracteres especiales**, como la ñ, ni tildes). Este directorio *src contiene cuatro ejercicios a resolver*.

El examen puede ser resuelto utilizando cualquier editor de texto plano y su correspondiente compilación y ejecución en el sistema operativo Linux. **Al profesor sólo se le subirán los códigos fuente sin incluir el ejecutable.** Aunque a continuación se detallan los ejercicios y el qué hay que realizar, a lo largo de los ficheros aparece la palabra **TODO** que indicará lo mismo.

Reglas para los ejercicios

- Usar el calificador *const.*

- Usar las directivas #ifndef #define #endif.

- No modificar nada de los archivos main, sólo descomentar y comentar.

- Comprobar los TODO.

- Ante cualquier duda, contactar con el profesor.

Hay que entregar:

- **En Moodle el directorio completo en formato .zip con el nombre "*AlternativaBloqueI-ApellidosNombre.zip*" (poner los apellidos y nombre del alumno).**
- **Al profesor un folio con la firma del alumno que se indica en esta misma página, y la resolución de los apartados a), b), c) y g) del ejercicio2, y el b) del ejercicio 3.**

Antes de darle a **enviar** los documentos en Moodle, **llamar al profesor** para comprobar que todo es correcto. **No se aceptarán envíos sin la supervisión del profesor**.

Confirmo que he leído las instrucciones del examen. Firma y hora de entrega:

Apellidos, Nombre:_____ DNI: _____

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 1 **Tiempo Estimado: 30minutos** **Puntuación: 3.5 puntos**

Dada la clase *Videojuego* completamente desarrollada (archivos *videojuego.h* y *videojuego.cpp*) que viene definida por un nombre (*string*), precio (*float*) y género *(string)*, **se pide implementar fuera de la clase las siguientes funciones (*cabecera.h y cabecera.cpp*):**

a) Dada una **lista** de videojuegos, devolver el valor medio de los videojuegos por encima de un precio umbral **(0.75 puntos). getValorMedio(...)**

b) Dado un **conjunto** de videojuegos, devolver una **lista** de dos elementos que contengan el videojuego de mayor y menor valor **(1 punto). getMayorMenorValor(...)**

c) Dada una **lista** de videojuegos, devolver un **map** cuya clave será el género, y el valor será un conjunto de videojuegos de dicho género **(0.75 puntos). getVideojuegosPorGenero(…)**

d) Dado un conjunto de videojuegos, devolver otro conjunto en el que se incremente en un porcentaje **p** aquellos videojuegos con un valor por debajo de la media **(1 punto). incrementarPrecio(...)**

Notas:

- Podrá incluir las funciones que necesite en los archivos *cabecera.h* y *cabecera.cpp*, pero se evaluará el funcionamiento de los apartados que se piden.

- El programa principal *main.cpp* no se tendrá que cambiar, sólo descomentar aquellas partes que se vayan completando para que compile.

Salida del programa:

```
APARTADO A)
El precio medio de los videojuegos por encima de 0€  es: 44.49€
El precio medio de los videojuegos por encima de 40€ es: 52.49€
El precio medio de los videojuegos por encima de 60€ es: 0€

APARTADO B)
(Super Mario Odyssey,59.99€,Aventura)
(Dark Souls III,19.99€,RPG)

APARTADO C)
[Accion]
(Assassin's Creed Valhalla,44.99€,Accion)
(Call of Duty: Modern Warfare,39.99€,Accion)
(God of War,49.99€,Accion)
[Aventura]
(Horizon Zero Dawn,39.99€,Aventura)
(Red Dead Redemption 2,49.99€,Aventura)
(Super Mario Odyssey,59.99€,Aventura)
(The Legend of Zelda: Breath of the Wild,59.99€,Aventura)
[RPG]
(Cyberpunk 2077,49.99€,RPG)
(Dark Souls III,19.99€,RPG)
(The Witcher 3: Wild Hunt,29.99€,RPG)

APARTADO D)
(Assassin's Creed Valhalla,44.99€,Accion)
(Call of Duty: Modern Warfare,43.989€,Accion)
(Cyberpunk 2077,49.99€,RPG)
(Dark Souls III,21.989€,RPG)
(God of War,49.99€,Accion)
(Horizon Zero Dawn,43.989€,Aventura)
(Red Dead Redemption 2,49.99€,Aventura)
(Super Mario Odyssey,59.99€,Aventura)
(The Legend of Zelda: Breath of the Wild,59.99€,Aventura)
(The Witcher 3: Wild Hunt,32.989€,RPG)
```

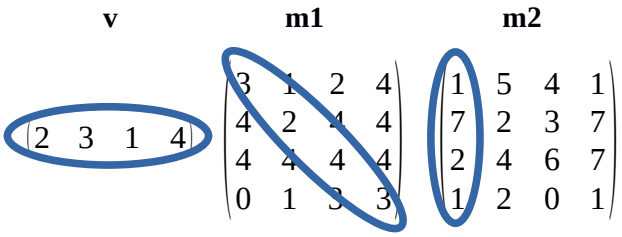
Apellidos, Nombre:_____ DNI: _____

EJERCICIO 2 **Tiempo Estimado: 30minutos** **Puntuación: 3.5 puntos**

Dada la misma clase que en el ejercicio 1, se pide crear un programa recursivo que trabaje con un vector v de *n* videojuegos, y dos matrices m1 y m2 de *nxn* videojuegos (matrices cuadradas). Se pretende sumar el precio de los videojuegos al cuadrado de v, restar el precio de los videojuegos de la diagonal principal de m1, y sumar los precios de los videojuegos de la primera columna de la matriz m2.

Con el ejemplo siguiente en el que sólo se muestran los precios de los videojuegos , el resultado sería:

30 – 12 + 11 = 29




```
Recursivo NO FINAL:
        Resultado 1: 29
        Resultado 2: 31
        Resultado 3: 113
        Resultado 4: 115
Recursivo FINAL:
        Resultado 1: 29
        Resultado 2: 31
        Resultado 3: 113
        Resultado 4: 115
ITERATIVO:
        Resultado 1: 29
        Resultado 2: 31
        Resultado 3: 113
        Resultado 4: 115
```

Se evaluarán los siguientes elementos:

a) Definición de la función **recursiva no final. (0.25 puntos)**

b) Definición de la función **recursiva final. (0.25 puntos)**

c) Definición de la **función iterativa. (0.25 puntos)**

d) Implementación de la **función recursiva no final. (0.75 puntos)**

e) Implementación de la **función recursiva final. (0.75 puntos)**

f) Implementación de la **función iterativa. (0.5 puntos)**

g) Cálculo de T(n) y cálculo de O(T(n)). **(0.75 puntos)**

Notas

- En un folio se debe entregar los apartados a, b, c, y g.

- Se debe entregar el directorio ejercicio2 con los apartados d, e y f.

- Además de la clase, este directorio cuenta con tres ficheros: *cabecera.h y cabecera.cpp* son los ficheros que tenéis que modificar (en ellos aparecen los TODO), mientras que *main.cpp* representa el programa principal para ver si se ha solucionado el ejercicio correctamente.

- Podrá incluir los métodos que necesite en los archivos *cabecera.h* y *cabecera.cpp*, pero se evaluará el funcionamiento de los métodos que se piden.

- El programa principal *main.cpp* no se tendrá que cambiar, sólo descomentar aquellas partes a medida que se van completando para que compile.

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 3 **Tiempo Estimado: 30minutos** **Puntuación: 2 puntos**

Dada la siguiente función recursiva múltiple, ya implementada:

$$f(n) = \begin{cases} 1 & \text{si } n < 4 \\ f(n-1)^2 - 4f(n-4) & \text{si } n \geq 4 \end{cases}$$

Se pide:

a) Implementación de la **función iterativa. (1 punto)**

b) Cálculo de T(n) y cálculo de O(T(n)). **(1 punto)**

Notas

- En un folio se debe entregar el apartado b.

- Se debe entregar el directorio ejercicio3 con el apartado a.

- Este directorio cuenta con tres ficheros: *cabecera.h y cabecera.cpp* son los ficheros que tenéis que modificar (en ellos aparecen los TODO), mientras que *main.cpp* representa el programa principal para ver si se ha solucionado el ejercicio correctamente.

- Podrá incluir los métodos que necesite en los archivos *cabecera.h* y *cabecera.cpp*, pero se evaluará el funcionamiento de los métodos que se piden.

- El programa principal *main.cpp* no se tendrá que cambiar, sólo descomentar aquellas partes a medida que se van completando para que compile.

Salida por pantalla

```
Recursivo:
        Prueba n=2: 1
        Prueba n=4: -3
        Prueba n=5: 5
        Prueba n=7: 437
Iterativo:
        Prueba n=2: 1
        Prueba n=4: -3
        Prueba n=5: 5
        Prueba n=7: 437
```

Progresión aritmética: $a_n = a_{n-1} + c$ **siendo c constante**

$$\sum_{i=1}^{n} a_i = \frac{n(a_1 + a_n)}{2}$$

Progresión geométrica: $a_n = ra_{n-1}$ **siendo** $r \neq 1$ **constante**

$$\sum_{i=1}^{n} a_i = \frac{a_n r - a_1}{r - 1}$$

Suma de cuadrados:

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 4 (Para Sobresaliente) **Tiempo Estimado: 20minutos** **Puntuación: 1.5 puntos**

Se pide implementar una plantilla de clase (template) que represente un Punto2D, cuyos atributos serán las coordenadas x e y. La plantilla debe funcionar correctamente con tipo de datos enteros y reales. Esto último se comprobará con el main.

a) Constructor con parámetros por defecto **(0.125 puntos).**

b) Observadores **(0.125 puntos).**

c) Modificadores **(0.125 puntos).**

d) Sobrecarga del **operador "="** como función miembro **(0.125 puntos)**.

e) Función mostrar, que mostrará un punto de la forma compacta (x,y) **(0.25 puntos).**

f) Función externa a la clase que reciba dos templates de Puntos2D y calcule la distancia euclídea entre ambos (**calcularDistanciaEuclidea**) **(0.75 puntos).**

Salida por pantalla

```
Probamos la plantilla para enteros
P1: (0,0)
P2: (3,4)
P3: (12,4)
Probamos la plantilla para reales
P4: (0,0)
P5: (9.5,4.5)
P6: (5.5,1.5)
La distancia Euclidea de los puntos 2 y 3 es: 9
La distancia Euclidea de los puntos 5 y 6 es: 5
```

Apellidos, Nombre:_____ DNI: _____