**Normas para la realización del examen:** **Duración: 2.5 horas**

- **El único material permitido durante la realización del examen es un bolígrafo azul o negro.**
- **Debe disponer de un documento oficial que acredite su identidad a disposición del profesor.**
- **No olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**

---

## ◁ Ejercicio 1 ▷ Pentagonal [3 puntos]

Un entero se dice que es pentagonal si puede obtenerse a partir de la fórmula $\frac{n(3n-1)}{2}$ para algún valor de $n > 0$. Los primeros 10 números pentagonales son 1, 5, 12, 22, 35, 51, 70, 92, 117 y 145. Se dice que una pareja de números pentagonales $p_i, p_j$ es metapentagonal si la media (entera) entre ambos $\frac{p_i+p_j}{2}$ y la diferencia de ambos en valor absoluto $abs(p_i - p_j)$ también son números pentagonales. Observe que ambos valores son **enteros** mayores que cero y menores que el máximo entre $p_i$ y $p_j$.

Se pide encontrar la pareja de números metapentagonales más alejada entre sí, es decir, aquella pareja $(p_i, p_j)$ cuya diferencia en valor absoluto $abs(p_i - p_j)$ sea máxima.

El programa leerá un entero $n$ que será el número de enteros pentagonales a considerar (imponga en la lectura que $0 < n <= 1000$) e imprimirá la pareja de números metapentagonales más lejana.

Puede suponer que no hay restricciones de memoria. Por contra, debe priorizar la rapidez de ejecución del programa.

---

## ◁ Ejercicio 2 ▷ Triángulo de Pascal [3.5 puntos]

En la figura 1.A se muestra un *Triángulo de Pascal* (o *Tartaglia*). Se puede observar que, exceptuando los números 1 que siempre están en los extremos, cada número es igual a la suma de los dos números que tiene justo encima.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fila 1 | | | | | | 1 | | | | | | | |
| Fila 2 | | | | | 1 | | 1 | | | | | | |
| Fila 3 | | | | 1 | | 2 | | 1 | | | | | |
| Fila 4 | | | 1 | | 3 | | 3 | | 1 | | | | |
| Fila 5 | | 1 | | 4 | | 6 | | 4 | | 1 | | | |
| Fila 6 | 1 | | 5 | | 10 | | 10 | | 5 | | 1 | | |
| Fila 7 | 1 | | 6 | | 15 | | 20 | | 15 | | 6 | | 1 |

A

| **SecuenciaEnteros** |
|---|
| - static const int TAMANIO = 100 |
| - int vector_privado[TAMANIO] |
| - int total_utilizados |
| + SecuenciaEnteros() |
| + int Capacidad() |
| + int TotalUtilizados() |
| + void Aniade(int valor) |
| + int Elemento(int indice) |
| + void Modifica(int indice,int nuevo)) |

B

Table 1: A) Triángulo de Pascal con 7 filas. B) Clase `SecuenciaEnteros`. Métodos que **NO hace falta implementar**

Se desea implementar la clase `TrianguloPascal`. Para usar eficientemente el espacio se decide emplear un objeto de la clase `SecuenciaEnteros` para guardar los valores del triángulo (dispone de la implementación ya terminada de la clase `SecuenciaEnteros` mostrada en la figura 1.B). La parte privada de la clase `TrianguloPascal` será:

```
SecuenciaEnteros valores;
int num_filas;
```

Por ejemplo, para el triángulo mostrado en la figura anterior, el campo `num_filas` valdrá 7 y la secuencia `valores` contendrá los valores: 1, 1, 1, 1, 2, 1, 1, 3, 3, 1, 1, 4, 6, 4, 1, 1, 5, 10, 10, 5, 1, 1, 6, 15, 20, 15, 6 y 1. Observe que los datos de la fila $f$ ($f \geq 1$) del triángulo empiezan en la posición $p = \frac{f(f-1)}{2}$ de la secuencia ($p \geq 0$).

La clase debe tener la siguiente funcionalidad:

1. Constructor sin argumentos. Crea y rellena un triángulo de Pascal con las **dos primeras** filas.
2. Métodos de consulta: a) `NumFilas`, para saber cuántas filas tiene el triángulo, y b) `NumCasillas`, para saber cuántas casillas tiene una fila dada del triángulo.

3. Escriba el método `AniadeFilas` para añadir al triángulo un número de filas dado.

4. Escriba el método `ObtenerFila` para obtener todos los valores de una fila `f` dada. Los valores se obtienen en un objeto de la clase `SecuenciaEnteros`. Si la fila solicitada no estuviera calculada, se calcularían (y añadirían al triángulo) tantas filas como fueran necesarias.

5. Escriba el método `Valor` para obtener el valor que ocupa la casilla `c` en la la fila `f`. Si la fila solicitada no estuviera calculada, se calcularían (y añadirían al triángulo) tantas filas como fueran necesarias.

Escriba un programa completo que lea un valor positivo `n` y cree un triángulo de Pascal con `n` filas. A continuación mostrará los valores del triángulo, fila a fila. Escriba todos los métodos (públicos o privados) adicionales que pudiera necesitar. Escriba también las precondiciones aplicables a los datos y métodos.

---

## ◁ Ejercicio 3 ▷ Imagen Color [3.5 puntos]

Un **píxel de color** es una terna $(x, y, z)$ con $0 \leq x, y, z \leq 255$ donde $x$ es el nivel de rojo, $y$ es el nivel de verde y $z$ es el nivel de azul. Una **imagen en color** es una matriz de píxeles de color. Podemos representar un píxel y una imagen mediante las siguientes clases.

| **PixelRGB** |
|---|
| *Datos miembro privados:* |
| - `char rojo;`<br>- `char verde;`<br>- `char azul;` |
| *Métodos públicos:* |
| + ..... |

| **ImagenRGB** |
|---|
| *Datos miembro privados:* |
| - `static const int MAX_DIM = 500;`<br>- `int ancho;`<br>- `int alto;`<br>- `PixelRGB imagen[MAX_DIM][MAX_DIM];` |
| *Métodos públicos:* |
| + ..... |

donde `imagen` es una matriz de objetos de la clase `PixelRGB` en la que la posición (0,0) se corresponde con el píxel en la esquina superior izquierda de la imagen. Se pide:

1. Un constructor de la clase `PixelRGB` que inicialice un píxel a un color RGB dado. Un constructor de la clase `ImagenRGB` que, dado un entero $1 \leq d \leq$ `MAX_DIM` y un `PixelRGB` $p$, cree una imagen *cuadrada* y *plana*: crea un objeto `ImagenRGB` de dimensiones $d \times d$ (cuadrada) donde todos los píxeles tienen el mismo color (plana) $p$.

2. Un método de la clase `ImagenRGB` que calcule y devuelva un **histograma** de una de las capas de color (R, G ó B) de la imagen. El método será de la forma:

   ```
   SecuenciaEnteros Histograma (char color);
   ```

   donde `color` será `R`, `G` ó `B` y la secuencia devuelta contendrá exactamente 256 casillas.

   Así, la ejecución de `paisaje.Histograma('R')` devuelve siempre una secuencia de 256 datos `int` (en definitiva, un objeto de la clase `SecuenciaEnteros`) con la frecuencia absoluta de cada valor en la capa `R` de `paisaje`: la casilla `0` de la secuencia contiene el número de veces que aparece el valor `0` en la capa 'R', la casilla `1` de la secuencia contiene el número de veces que aparece el valor `1` en la capa 'R', y así sucesivamente.

3. Escribir un método que concatene horizontalmente dos imágenes. El resultado es otra imagen mayor que las que intervienen: el ancho es la suma de los anchos de las dos imágenes, y el alto es el mayor entre las dos imágenes. Los "huecos" se rellenan con píxeles en negro (en RGB sería (0,0,0)).

Se deben implementar todos los métodos, de ambas clases, que sean necesarios para realizar los puntos anteriores.