Diseñe los siguientes programas en el lenguaje de programación Java, teniendo en cuenta los siguientes parámetros:

- Los programas deben ser desarrollados exclusivamente con la técnica descrita al inicio.
- Los tres programas deben ser integrados en un menú principal que permita acceder a cada uno de ellos.
- Los programas deben ser flexibles (dinámicos). De ninguna manera se aceptarán programas con datos "quemados en código", o funcionando exclusivamente para algunos casos particulares.
- Los programas podrán ser desarrollados por grupos de máximo 2 personas y serán sustentados sólo por una de ellas, escogida al azar.

Programas:

1. Dividir y Vencer

Desarrollar un algoritmo dividir y vencer para calcular el producto de dos matrices cuadradas triangulares superiores, siendo el número de filas y columnas potencia de 2.

Recordatorio: una matriz se dice que es triangular superior si todos los elementos que están por debajo de la diagonal principal valen 0. Se dice que es triangular inferior si todos los elementos encima de la diagonal principal valen 0. Ejemplo de matriz triangular superior:

1	3	5
0	2	4
0	0	2

Entradas del Programa:
Tamaño de las dos matrices
Salidas del programa
Matriz resultante de la multiplicación

Nota: El programa deberá cargar aleatoriamente los datos de cada una de las matrices de entrada.

2. Programación dinámica

Sobre un río hay n embarcaderos. En cada uno de ellos se puede alquilar un bote que permite ir a cualquier otro embarcadero río abajo (es imposible ir río arriba). Existe una tabla de tarifas que indica el coste del viaje del embarcadero i al j para cualquier embarcadero de partida i y cualquier embarcadero de llegada j más abajo en el río (i < j). Puede suceder que un viaje de i a j sea más caro que una sucesión de viajes más cortos, en cuyo caso se tomaría un primer bote hasta un embarcadero k y un segundo bote para continuar a partir de k. No hay coste adicional por cambiar de bote.

El problema consiste en diseñar un algoritmo eficiente que determine el coste mínimo de ir de un punto i a un punto j, indicando cuáles embarcaderos se utilizaron.

Entradas del Programa:

Número de embarcaderos, con los costes entre sí que serán cargados de manera aleatoria (tarifas).
Embarcadero inicial y embarcadero final.

Salidas del programa

Una tabla dinámica que evidencie el funcionamiento del algoritmo
El costo mínimo para ir del embarcadero inicial al final
Identificación de los embarcaderos que se utilizan para encontrar el coste.

3. Vuelta atrás

Una matriz bidimensional nxn puede representar un laberinto cuadrado. Cada posición contiene un entero no negativo que indica si la casilla es transitable (1) o no lo es (0). Las casillas [1,1] y [n,n] corresponden a la entrada y salida del laberinto y siempre serán transitables. Generando una matriz con el laberinto, de forma aleatoria, diseñar un algoritmo que encuentre un camino, si existe, para ir de la entrada a la salida.

1	0	1	0	1
1	1	1	1	1
1	0	1	1	0
0	0	0	1	0
0	0	0	1	1

Entradas del Programa:

Tamaño del laberinto (#filas y #columnas)

Salidas del programa

Una muestra paso a paso del camino recorrido por el algoritmo, de tal forma que evidencie la búsqueda de la salida, utilizando una estrategia vuelta atrás