Problema 1 (2 puntos).

En la gestión de la red social PiensaQue (www.piensaque.com, donde puedes escuchar qué piensan tus contactos) quieren determinar el grado de conexión entre sus usuarios. El grado de conexión es la cantidad de grupos de usuarios conectados entre sí dividido por el total de usuarios. Si este valor se acerca a 1 es que todos los usuarios se conocen, bien directamente o bien a través de amigos en común (o amigos de amigos de un amigo, etc.) mientras que si se aproxima a 0 quiere decir que los usuarios están aislados y prácticamente sin contactos comunes.

Entre los miles de usuarios de PiensaQue se escoge una población aleatoria de cien identificadores para obtener su grado de conexión. Cada uno de esos cien identificadores representa a un usuario, del que se además se tienen sus contactos almacenados en una lista. Por ejemplo, se podría tener el siguiente caso de seis usuarios (en cursiva) y sus contactos (separados por comas):

Antonio: Bea, Carlos, Emma	*Carlos:* Antonio, Emma	*Emma:* Bea
Bea: Emma, Carlos, Antonio	*David:* Fernando	*Fernando:* NINGUNO

Aunque Emma solo tiene por contacto a Bea, a través de ella está relacionada con Carlos y con Antonio. Por otra parte, Fernando y David también tienen una conexión entre ellos (David tiene a Fernando entre sus contactos, aunque el recíproco no se cumpla). Sin embargo, los dos grupos no tienen nexos en común.

En este ejemplo el grado de conexión sería 2 grupos entre 6 usuarios, aproximadamente el valor 0’33.

Diseñar un algoritmo **voraz** que, teniendo como datos los usuarios elegidos (cada uno de ellos con su identificador y la lista de sus contactos), obtenga el grado de conexión entre ellos.

Requisitos de entrega:

- Entregar una memoria en la que se explique la solución, justificando por qué se ha optado por la implementación llevada a cabo.
- Código fuente.
- Fichero de entrada ‘ejemplo_voraz.txt’
- Ejecutable que se pueda probar insertando el fichero ejemplo, “ejemplo_voraz.txt”, y devuelva un fichero de salida con el número de grupos, los integrantes de estos grupos y el grado de conexión. El ejecutable debe funcionar en cualquier entorno Windows. El fichero de ejemplo debe leerse de la misma carpeta donde se encuentra el ejecutable.

Problema 2 (1 punto).

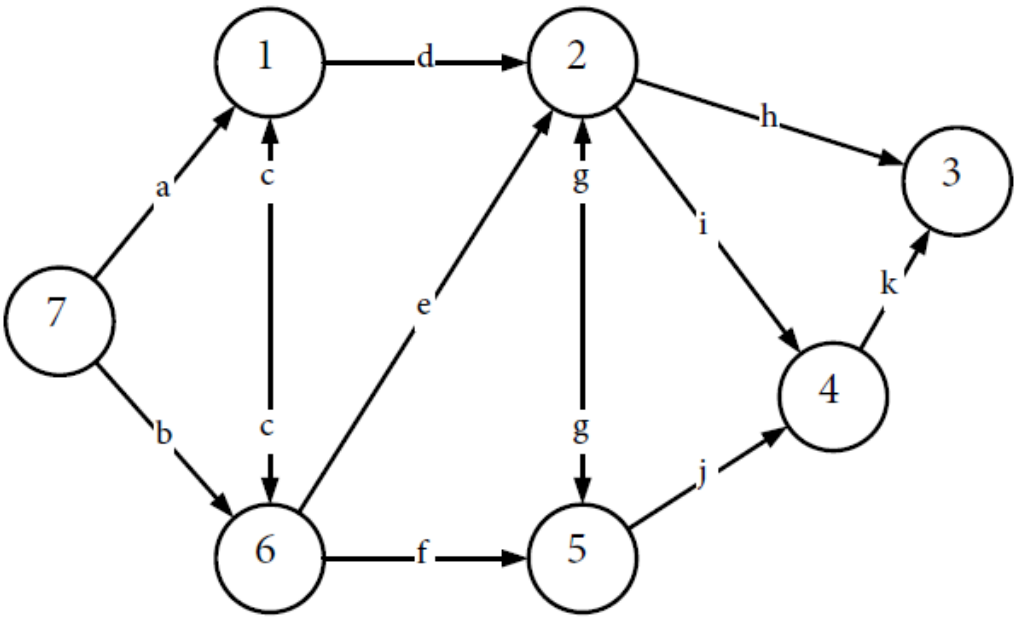
Calcular la nota media de un alumno en un curso formado por n asignaturas usando la técnica **divide y vencerás**, siendo n potencia de 2. Modificar el algoritmo para permitir valores de n que no sean potencia de 2. Calcular la eficiencia y determinar si mejora la que se obtiene respecto a una simple versión iterativa.

Requisitos de entrega:

- Entregar una memoria en la que se explique la solución, justificando por qué se ha optado por la implementación llevada a cabo.
- Código fuente.
- Fichero de entrada 'ejemplo_dyv.txt'
- Ejecutable que se pueda probar insertando el fichero ejemplo, "ejemplo_dyv.txt", y devuelva un fichero de salida con el valor medio obtenido. Se debe mostrar el valor medio que se obtiene en todas las etapas recursivas del proceso divide y vencerás. El ejecutable debe funcionar en cualquier entorno Windows. El fichero de ejemplo debe leerse de la misma carpeta donde se encuentra el ejecutable.

Problema 3 (0.75 puntos)

Dado el siguiente grafo:



Se pide:

a) Generar 11 números aleatorios entre 1 y 10.
b) Sustituir las letras de los ejes del grafo por los números aleatorios generados.
c) Aplicar el algoritmo de Floyd y obtener la matriz de distancias mínimas del grafo. Mostrar todas la iteraciones intermedias del algoritmo.

Problema 4 (2 puntos).

Bruce Springsteen va a terminar su gira mundial de este año dando el último concierto en Abecelandia, ciudad famosa por sus bellas plazas y que puede que conozcas, por lo que un camión cargado de instrumentos, aparatos de sonido, bengalas, soportes y tuercas ha entrado en la ciudad y se dirige al estadio para montar el escenario y los fuegos artificiales. Repentinamente, una de las ruedas delanteras explota y deja parcialmente inutilizado el sistema de dirección del camión, impidiéndole realizar giros hacia la izquierda. Debido a su tamaño y peso, el camión no puede ir marcha atrás (así que no puede retroceder) ni cambiar de sentido en la misma calle (es decir, no puede dar un giro de 180º sin cambiar de calle) así que, de momento, el camión solo puede circular en línea recta y realizar giros de 90º a la derecha. Afortunadamente el conductor dispone de un plano de Abecelandia como el siguiente, donde el punto 1 es la posición actual del camión, el punto 2 es el estadio al que debe ir, y cada posición está marcada como circulable (si está vacía) o no circulable (si está en negro). Puede suponerse que inicialmente el camión está orientado hacia la izquierda, y que el mapa tiene tamaño FxC. Por tanto, en cada posición del mapa el camión puede continuar avanzando (suponiendo que la siguiente casilla esté libre) o girar 90º a la derecha (una vez más, suponiendo que esa posición sea circulable). Como no se sabe si el resto de ruedas aguantarán hasta llegar al estadio, se quiere reducir al máximo la cantidad de giros para no sobrecargar demasiado los neumáticos que aún funcionan. Implementar un algoritmo con metodología **Backtracking**, eficiente en la medida de lo posible, que teniendo como datos los tamaños en filas F y columnas C del mapa, el mapa MFxC y las coordenadas de los puntos 1 y 2, encuentre el camino que lleva de 1 a 2 con la menor cantidad de giros hacia la derecha.

IMPORTANTE: Es imprescindible cumplir con los requisitos de entrega y los formatos de entrada.

Formato de entrada: un fichero “ejemplo_backtracking.txt”, con el siguiente formato:

- Línea 1: Número de filas (F).
- Línea 2: Número de columnas (C).
- Línea 3: Descripción de la primera fila (C caracteres separados por tabulaciones).
- Línea 4: Descripción de la segunda fila (C caracteres separados por tabulaciones).
- …
- Línea 2+F: Descripción de la última fila (C caracteres separados por
- tabulaciones).

Descripción de los caracteres posibles:

- ‘1’: Inicio en el mapa.
- ‘2’: Final en el mapa.
- 0’: Casilla libre en el mapa, por la que se puede circular.
- ‘x’: casilla ocupada en el mapa, por la que no se puede circular.

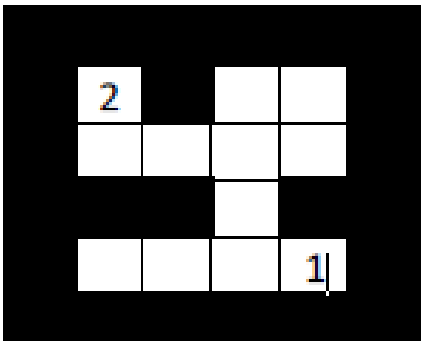
Requisitos de entrega:

- Entregar una memoria en la que se explique la solución, justificando por qué se ha optado por la implementación llevada a cabo.

- Código fuente.
- Ejecutable que se pueda probar insertando un fichero cualquiera de entrada, “ejemplo_backtracking.txt”, y devuelva un fichero de salida con todas las soluciones encontradas, así como la óptima. Se debe especificar el recorrido concreto para cada solución (por ejemplo, indicando si en cada casilla por la que se pasa se sigue recto o se gira hacia la derecha). El ejecutable debe funcionar en cualquier entorno Windows. El fichero de ejemplo debe leerse de la misma carpeta donde se encuentra el ejecutable.

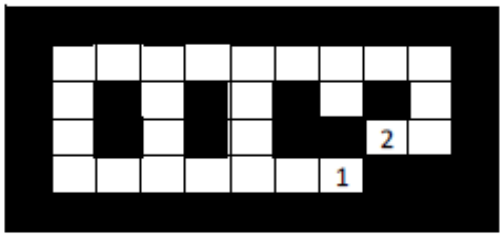
Ejemplos:

“Ejemplo 1.txt”:



2	X	0	0
0	0	0	0
X	X	0	X
0	0	0	1

“Ejemplo 2.txt”**:**



0	0	0	0	0	0	0	0	0
0	X	0	X	0	X	0	X	0
0	X	0	X	0	X	X	2	0
0	0	0	0	0	0	1	X	X

Problema 5 (0.75 puntos)

El siguiente cuadro corresponde a un problema de asignación:

	1	**2**	**3**	**4**
a	a_{11}	a_{12}	a_{13}	a_{14}
b	a_{21}	a_{22}	a_{23}	a_{24}
c	a_{31}	a_{32}	a_{33}	a_{34}
d	a_{41}	a_{42}	a_{43}	a_{44}

Se pide:

a) Generar 16 números aleatorios entre 5 y 35.
b) Sustituir los valores a_{ij} por los números generados.
c) Resolver el correspondiente problema de asignación utilizando ramificación y poda.