CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE SISTEMAS ELECTRONICOS
ACADEMIA DE PROGRAMACION

Materia	Estructuras de Datos	Carrera	ING. EN SISTEMAS
Profesor	Dra. BLANCA ESTRADA RENTERIA	Semestre	3°
PROYECTO FINAL			
Observaciones:	Todos los proyectos deberán cumplir con las instrucciones generales y debe de hacer una presentación tipo ejecutiva donde presente su proyecto. Máximo número de integrantes de equipo 4		

Instrucciones Generales:

Deberá entregar el código fuente y el compilado del proyecto en una carpeta comprimida en aula virtual, la carpeta contará con todos los archivos necesarios para su ejecución y sin virus.

Además debe realizar un **manual de usuario** que muestre el funcionamiento de su proyecto, tome capturas de pantalla y describa de manera clara como se realiza cada tarea de su aplicación.
Así mismo, puede incluir todos aquellos documentos que crea necesarios como investigación adicional.

Entregar impresa y engargolada la documentación con lo siguiente:

- Portada
- Resumen descriptivo de su proyecto haciendo énfasis en las fortalezas y debilidades de su aplicación realizada.
- Temas investigados para la realización del proyecto.
- Bitácora de trabajo por cada uno de los integrantes del equipo.
- Conclusiones por cada uno de los integrantes del equipo.
- Referencias consultadas.
- Manual de usuario

Además, el documento elaborado debe estar incluido en pdf dentro del comprimido que entrega en plataforma, con su proyecto.

Proyecto copiado de Internet o de alguna otra fuente causará anulación de la calificación final del curso, así como proyectos con errores de compilación o que no se realice lo que se pide.

TEMAS A EVALUAR:
Se evaluará que el proyecto **no tenga errores de compilación y/o sintaxis, errores de ejecución**. Además, debe cumplir con todos los temas que marca el programa de la materia, como se describa en su proyecto.

A continuación se sugiere como realizar los el documento a entregar en pdf :

1. Portada
    a) Imagen con título de la UAA
    b) Título del proyecto
    c) Nombre de la materia
    d) Nombres de los integrantes
    e) Carrera, semestre y grupo
    f) Nombre del profesor
    g) Fecha de entrega

2. Resumen descriptivo
    a) Fortalezas
    b) Debilidades

    Estos puntos los deben redactar con sus propias palabras, indiquen que fortalezas tiene su proyecto y cuales debilidades, justifiquen sus comentarios.

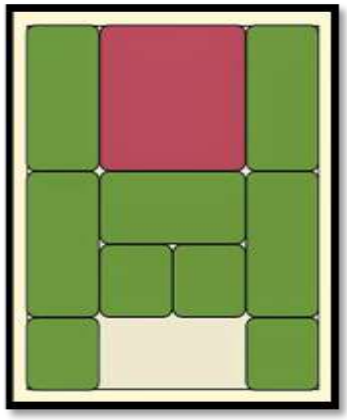
3. Temas Investigados para la realización del proyecto, si realizo alguna investigación en particular.
4. Bitácora de trabajo
   Hacer una tabla donde incluyan el nombre del integrante del equipo, actividades realizadas y tiempo (días) que se llevó en realizar dicha actividad (sean honestos y escriban lo que hizo cada uno)
5. Conclusiones
   Con sus palabras expliquen de forma breve que fue lo que aprendieron con el desarrollo del proyecto y que suma a su formación como futuro ingeniero en sistemas computacionales.
6. Bibliografía o Referencias consultadas
   Escribir la bibliografía que fue consultada para realizar el proyecto (libros, páginas de Internet, etc.)
   Recuerden que un formato adecuado para citar fuentes de libros, revistas, páginas web, periódicos, etc.
7. **manual de usuario** que muestre el funcionamiento de su proyecto, tome capturas de pantalla y describa de manera clara como se realiza cada tarea de su aplicación.

En cuanto a la realización de código, tome en cuenta lo siguiente:

- Debe aplicar todos temas vistos en el curso: pilas, colar, lista, arboles, grafos, búsquedas y ordenaciones.
- Utilizar los menús necesarios para acceder a cada parte del programa de manera clara y debidamente validados.
- Validar las capturas (que no existan errores de ejecución, es decir, que no “truene” el programa).
- Cada sección de código del programa debe estar perfectamente bien comentada, es decir, cada función o clase debe tener comentarios acerca de lo que se realiza.
- Los nombres de las funciones o clases deben ser de acuerdo a la funcionalidad que implementa.
- Si su proyecto incluye gráficos se le agregaran **hasta** 2 puntos extra, eso dependiendo de la calidad del uso de los mismo.

NOTA IMPORTANTE: Todos los proyectos deben contar con ayuda, la cual se almacena en un archivo de texto, mismo que debe ser capaz de desplegar en su respectivo proyecto, así mismo deberán realizar una presentación ejecutiva para la entrega/presentación de su proyecto.

Proyecto Klotski



Klotski

Klostsky es un rompecabezas de piezas deslizantes en dónde el objetivo es mover una pieza específica a un lugar determinado (Sacarla de la cárcel; otro nombre adjudicado al juego).

El juego se compone de un tablero ocupado por piezas y muros, y el objetivo del juego es desplazar una de las piezas de su posición inicial a una posición objetivo.

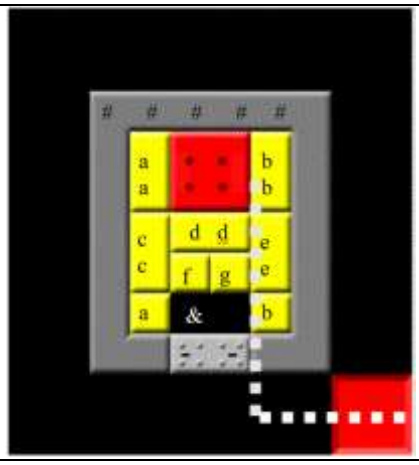
Reglas

1. El tablero está dividido en cuadrados, de modo que cada elemento que contiene ocupa uno o varios cuadrados contiguos (no se consideran contiguos cuadrados que se toquen en un único punto).
2. Los diferentes elementos del tablero son:
    - Piezas
    - Una pieza singular
    - Muros
    - Puertas
    - Posición objetivo
3. La pieza singular encajará completamente en la posición objetivo.
4. Las piezas (y sólo ellas) se pueden desplazar por el tablero, horizontal o verticalmente, pero no pueden moverse por encima de otras piezas o de muros. Cualquier pieza puede moverse o colocarse encima de la posición objetivo. Únicamente la pieza singular puede moverse o colocarse encima de las puertas.
5. **El programa leerá el reto a resolver desde un archivo de texto. El archivo a leer se llamará nivel_N.txt. (Opción: Cargarlo desde la interface del juego // programación)**

Formato de archivo.

1. La primera línea es el nombre del nivel. El nombre contendrá como mucho 40 caracteres.
2. La segunda línea contiene dos números naturales, el primero es el ancho del tablero en cuadrados, y el segundo la altura del tablero en cuadrados. En principio no hay límite en ambos números.
3. Las siguientes líneas, tantas como la altura del tablero en cuadrados, representarán el tablero. En cada línea, cada carácter representará qué elemento ocupa el cuadrado correspondiente del tablero (por tanto, en el tablero inicial no puede haber elementos solapados)
4. Los diferentes elementos se representan de la siguiente forma:
    - Pieza singular: *
    - Resto de las piezas: a - z (Letra minúscula ASCII)
    - Muro: #
    - Puerta: -
    - Posición objetivo: .
    - Cuadrado vacío: ‘ ’ o & (espacio en blanco)

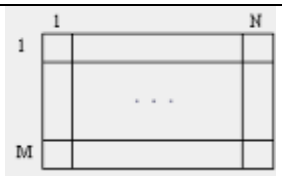
Ejemplo:



5. **La solución debe cargar nuevos tableros y resolverlo sin necesidad de recompilar el ejercicio.**
    a. Al cargar el tablero el programa deberá dibujarlo en forma gráfica y generar la(s) solución(es) posible(s).
    b. La pantalla del juego contendrá al menos:
        i. El tablero vacío.
        ii. Método de carga del archivo (reto).
        iii. Dibujar el tablero completo.
        iv. Botón para general solución.
        v. Iniciar recorrido.
    c. Las piezas y los muros se representarán de modo que aparezcan conexos, inclusive si ocupan varios cuadrados del tablero.
    d. Cada pieza diferente que se mueve se considera un movimiento. Al mover una pieza que acaba de ser movida se considera que es una continuación del movimiento anterior, y no cuenta como movimiento.
    e. Al final deberá indicar cuantos movimientos realizo para solucionar el reto, así como el camino recorrido para la solución

Proyecto: Tom y Jerry

Como es usual, el gato Tom persigue a Jerry por las habitaciones de una casa sin ningún éxito, pues Jerry conoce todos los recovecos de la mansión. Harto de esta situación, Tom adquiere un PC con la intención de construir un programa capaz de encontrar el camino más corto hasta su ansiada presa sorteando todos los obstáculos (paredes, armarios, etc.) que pueda encontrarse. Además, para asegurar el éxito de la cacería, Tom desea atacar a Jerry cuando éste duerma.

Representaremos la casa por una superficie rectangular de M filas y N columnas, dividida en casillas unitarias:	

Tom tan solo puede moverse en horizontal o vertical dentro de la habitación, nunca en diagonal. Supondremos que tanto Tom como Jerry ocupan una casilla dentro de esta superficie. Los obstáculos de la habitación serán rectángulos de coordenadas válidas dentro de la superficie, y se representarán con su vértice superior izquierdo y el inferior derecho. Existe la posibilidad que el rectángulo sea en realidad una recta (horizontal o vertical; por ejemplo, las coordenadas (3, 4) y (3, 6) definen una recta horizontal) o incluso un punto (cuando las dos coordenadas son iguales).

Fase1:
Construir un programa que procese un archivo de entrada, de nombre "TOM1.DAT", que contenga la configuración inicial de la casa, y que la dibuje si es correcta. Los casos de error que consideramos serán: (E0) M o N (o ambas) son igual a 0. (E1) Tom o Jerry no se encuentran en coordenadas válidas de la superficie de la casa. (E2) Tom y Jerry están en la misma casilla. (E3) Algún obstáculo no está situado en coordenadas válidas de la superficie. (E4) Los vértices que representan un obstáculo no cumplen una relación válida entre sí. (E5) Dos o más obstáculos se solapan. (E6) Tom o Jerry están en una casilla ocupada por un obstáculo.

El formato del fichero "TOM1.DAT" es el siguiente:
Línea 1: valores de M y de N, separados por un único espacio en blanco (tanto M como N se representan con un único dígito)
Línea 2: cuatro valores naturales, separados por un único espacio en blanco, representados cada uno de ellos por un dígito. Los dos primeros valores representan la posición inicial de Tom (fila - columna) y los dos siguientes la de Jerry.
Líneas siguientes: cada línea se corresponde a un obstáculo y contiene cuatro valores naturales, separados por un único espacio en blanco, y representados cada uno de ellos por un dígito. Los dos primeros valores representan el vértice superior izquierdo del obstáculo (fila - columna) y los dos restantes el vértice inferior derecho.
La salida debe almacenarse en el fichero "TOM1.RES". En caso de encontrarse algún error, el fichero contendrá una única línea con el mensaje "ERROR Ex", siendo la 'x' un número entre 0 y 6 que identifica el tipo del primer error que se encuentre en la entrada. Si hubiese más de un error a la vez, se devolverá aquel con código menor; por ejemplo, si el primer error lo provoca un obstáculo que se solapa con alguno de los anteriores y al mismo tiempo ocupa la casilla de Tom o de Jerry, el código a devolver sería E5, este error debe también mostrarse en pantalla y regresarlo al menú principal para poder cargar otro archivo.
En caso de no encontrarse ningún error, el fichero contendrá M líneas de N caracteres cada una de ellas, siendo cada carácter el contenido de una casilla, que será 'o' para las casillas vacías, 'x' para las casillas que forman parte de un obstáculo, 'T' para la casilla ocupada por Tom y 'J' para la casilla ocupada por Jerry.
A continuación, ofrecemos algunos ejemplos de entrada errónea y el mensaje que debe guardarse en el fichero de salida:

TOM1.DAT	TOM1.DAT	TOM1.DAT
3 3 2 2 3 3 1 2 2 1	3 3 2 2 3 2 1 2 2 3	3 3 3 2 2 3 1 1 2 2 1 2 2 3
TOM1.RES	TOM1.RES	TOM1.RES
ERROR E4	ERROR E6	ERROR E5

En cambio, las entradas siguientes generan resultados correctos:

TOM1.DAT	TOM1.DAT
5 5 1 1 5 3 2 1 3 3 1 4 4 4	7 5 1 2 7 1 2 1 2 2 2 4 2 4 4 2 4 5 6 1 6 2 6 4 6 4
TOM1.RES	TOM1.RES
Tooxo xxxxo xxxxo ooooxo ooJoo	oTooo xxoxo ooooo oxxxx ooooo xxoxo Joooo

Fase 2

Su programa debe procesar una entrada que no contenga ningún error, debe **calcular el camino lo más corto posible que lleve de Tom a Jerry**. El programa leerá la información del fichero "TOM2.DAT", con el mismo formato que el fichero "TOM1.DAT" del apartado anterior, desplegara en pantalla la distribución de la casa además, la salida se almacenará en el fichero "TOM2.RES", que contendrá diferentes líneas con el resultado del algoritmo. En concreto, el fichero contendrá una línea para cada posición que forme parte del camino en el mismo orden en que aparecen en él, siendo pues la primera la posición de Tom y la última la posición de Jerry. Concretamente, cada línea contendrá un par de valores naturales, representados mediante un dígito y separados por un único espacio en blanco; el primer natural representa la fila y el segundo la columna, así mismo, debe indicar el camino que se tomo en la pantalla sobre la distribución de la casa marcada previamente. En caso de que no haya ninguna solución posible, el fichero "TOM2.RES" deberá contener una única línea con la palabra "INALCANZABLE" y este deberá mostrarse también en pantalla. En caso de que haya más de un camino mínimo, puede devolver cualquiera de ellos.
Así para las entradas del último ejemplo del apartado anterior, para el primero de ellos el fichero "TOM2.RES" debería contener una única línea con la palabra "INALCANZABLE". En cambio, para el segundo, al existir un único camino mínimo, el resultado sería el fichero "TOM2.RES" siguiente:

TOM2.RES

1 2
1 3
2 3
3 3
3 2
3 1
4 1
5 1
5 2
5 3
6 3
7 3
7 2
7 1

Fase 3:

Su programa debe tener una opción que devuelva todos los caminos que lleven de Tom a Jerry, sean lo largos que sean, y tales que no se pasa más de una vez por una misma casilla. Para minimizar el tamaño de la salida, nos limitaremos a decir cuántos caminos hay de cada longitud. La longitud de un camino se define como el número de posiciones que lo componen menos uno; en el último ejemplo del apartado anterior, el camino que aparece tiene longitud 13.

El programa leerá la información del fichero "TOM3.DAT", con el mismo formato que el fichero "TOM2.DAT" del apartado anterior. La salida se almacenará en el fichero "TOM3.RES", que contendrá diferentes líneas. Cada una de las líneas estará formada por dos valores naturales, representados mediante tantos dígitos como sea necesario y separados por un único espacio en blanco; el primer natural representa la longitud y el segundo el número de caminos existentes de esa longitud. No se debe incluir líneas para aquellas longitudes que no sean longitudes de algún camino de Tom a Jerry. Las líneas deben estar ordenadas por la longitud, estos caminos también deben ser presentados por pantalla.

La figura siguiente muestra un ejemplo de entrada y el resultado esperado para ella:

TOM3.DAT	TOM3.RES
4 4 1 1 4 4 3 2 3 3 2 3 2 3	6 2 8 2

Proyecto: Buque

La línea marítima "Titanic S.A.", cuya flota consta de un único buque, se ocupa del transporte de mercancías desde el puerto a su destino. Cada uno de los productos tiene un volumen (por motivos de conservación, las mercancías van siempre embaladas en cajas) y un precio de venta. Ante la convocatoria inminente de una huelga de estibadores, la línea decide efectuar un viaje extraordinario llenando el buque de manera que la mercancía que transporte sea lo más valiosa posible. Se pide construir un programa que, dada la información de las mercancías (volumen, precio y unidades disponibles), determine cuáles deben transportarse en el buque sin desbordar su capacidad (que será una información adicional sumistrada al programa). En caso que existan diversas combinaciones óptimas de mercancía con respecto al precio, se elegira aquélla que ocupe menos volumen; en este caso, sí puede suponerse que existe una única solución óptima al problema.

Entrada
Residente en el fichero de caracteres "BUQUE.DAT":
Línea 1: número N de tipos de mercancías, mediante uno o dos caracteres que representan un número entero entre 1 y 99.
Línea 2: capacidad del buque, mediante uno, dos, tres o cuatro caracteres que representan un número entero entre 1 y 9999.
Líneas de la 3 a la N+2: cada una de las líneas tiene el formato:

mercancía – volumen - coste unidades

donde mercancía es una palabra formada exclusivamente por letras minúsculas (y sin signos de puntuación, es decir, ni acentos ni similares) de máximo 20 letras, y los otros tres componentes son enteros representados mediante un número de dígitos que oscila entre 1 y 5 (es decir, el entero más grande representable es el 99999). Los componentes de la línea están separados por un único carácter blanco, y no existen blancos ni otro tipo de caracteres al principio o final de línea.

Fase 1:

Ud debera hacer un programa que una vez que lee un archivo correcto, depliegue en pantalla:

- El espacio disponible para colocar la mercancia.
- La mercancia que tiene que embarcarse
- Si por algun motivo de error en el archivo de entrada no hay espacio para embarcar la mercancia, debera mostrar un mensaje de error en ese sentido.

Fase 2:
Su programa debe procesar una entrada que no contenga ningún error, debe **calcular la forma mas optima de embarcar el mayor número de paquetes posibles**. Una vez realizado el cálculo, debe desplegara en pantalla la distribución de la mercancía dentro de los espacios asignados para tal fin, la salida se almacenará en el fichero "Buque.RES", que contendrá diferentes líneas con el resultado del algoritmo. En concreto, el fichero contendrá una línea para cada paquete embarcado en el mismo orden en que se realizó la selección del mismo. Concretamente, cada línea contendrá tres valores equivalentes a **mercancía – volumen - coste unidades**, mismas que se embarcarán, estas deberán tener el mismo formato con el cual fueron cargadas inicialmente.
. En caso de que no haya ninguna solución posible, el fichero BUQUE.RES deberá contener una única línea con la palabra "INALCANZABLE" y este deberá mostrarse también en pantalla.

Fase 3:

Su programa debe tener una opción que devuelva todas las formas posibles de embarcar la carga. Para minimizar el tamaño de la salida, nos limitaremos a decir cuantas formas diferentes existen.

Ejemplo de entrada
5
2000
patatas 350 2 7
judias 400 5 4
guisantes 1000 12 4
fresones 1100 17 3
arroz 600 8 1

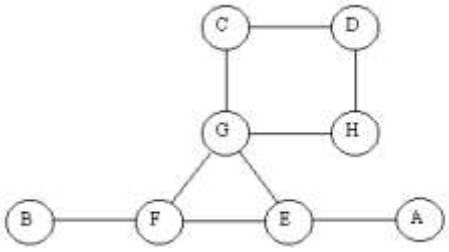
Ejemplo de salida

27 1900
fresones 1
judias 2

Proyecto: Redes interesantes

Existen varios ámbitos de aplicaciones informáticas que manejan el concepto de red topológica. Por ejemplo, cuando se modelizan redes de ordenadores, de carreteras, y similares. Sobre estas redes, nos puede interesar preguntar ciertas cosas: si dos ordenadores están conectados, cuál es el camino más corto para ir de una ciudad a otra, etc. En este problema, nos dirigimos a un problema concreto en este contexto, la búsqueda de subredes interesantes.

Definición
Sea una red R que contiene un conjunto V de elementos (por ejemplo, ordenadores) y un conjunto C de conexiones bidireccionales entre pares de elementos de V. Una subred interesante de R se define como un conjunto W de más de dos elementos talque W_i V y todo par de elementos de W está conectado por alguna conexión de C.
Por ejemplo, en la red que se muestra en la figura siguiente, existe una única subredinteresante, que aparece en el centro de la red (uniendo los nodos E, F y G).



Objetivo
Dada una red R, se quiere obtener todas sus subredes interesantes que sean maximales, es decir, que no formen parte de ninguna otra subred interesante.

Entrada
La entrada del programa consiste en una secuencia de líneas, que residen en unarchivo de texto (ASCII) con nombre RED.DAT, que tendrá el siguiente formato:

- La primera línea (la 1) contiene el número m de conexiones de la red. Puede suponer 1 <= m <= 99.
- Las m líneas siguientes contienen, cada una, dos letras mayúsculas (del alfabeto anglosajón, es decir, sin 'Ñ' ni 'Ç' pero con 'W') sin acentos separadaspor caracteres "espacio", que representan conexiones entre elementos de lared.

Fase 1:

Ud debera hacer un programa que una vez que lee un archivo correcto, depliegue en pantalla:

- La red definida por el.
- Si por algun motivo de error en el archivo de entrada, debera mostrar un mensaje de error en ese sentido.

Fase 2:

Su programa debe ser capaz de identificar las subredes dentro de la red principal y deplegar en pantalla:

- La red definida por el.
- El numero de subred interesante maximales encontradas.

- Idetificar visualmente cada una de las sub redes identificadas

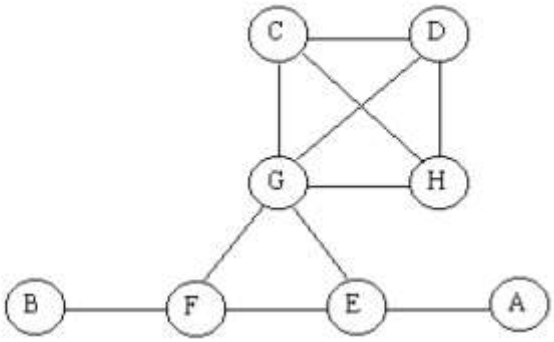
Fase 3:

Ademas, la salida del programa ha de grabarse en un archivo de texto (ASCII) con nombre RED.RES, que contendrá una línea por cada subred interesante maximal que contengala red, escribiendo los elementos que lo forman en orden alfabético separados por un único carácter "espacio". En la salida, las subredes se ordenan por número de elementos que contienen y, en caso de subredes igual de grandes, no se exige ningún orden adicional.

Ejemplo
Se muestra una entrada que representa una red que contiene dos subredesinteresantes maximales. En concreto, nótese que la subred formada por loselementos C, D, G y H contiene otras subredes interesantes pero que, al no sermaximales, no aparecen en el resultado.

Ejemplo de entrada

```
11
E F
F B
E A
H G
C D
G E
C H
F G
H D
G D
C G
```



Ejemplo de salida

```
C D G H
E F G
```