UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
2021-1 - Tarea Integradora 2

Objetivos

Unidad 3: Estructuras Lineales Enlazadas

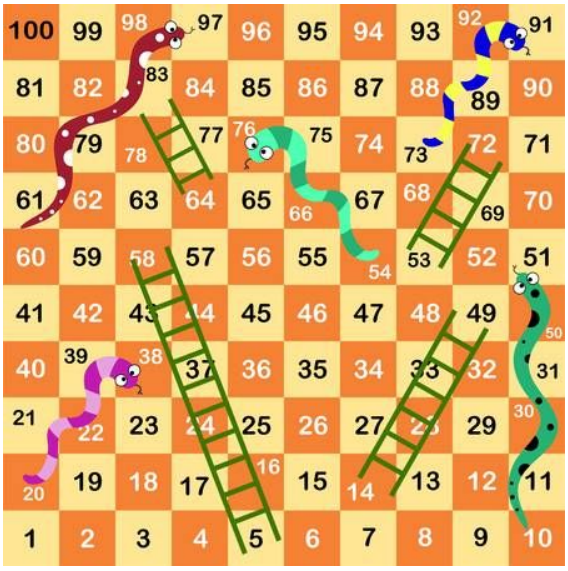
OE3.1 Utilizar estructuras enlazadas de objetos para modelar grupos de atributos no primitivos de tamaño flexible.
OE3.2 Escribir los algoritmos necesarios para manipular estructuras lineales que almacenan sus elementos enlazándolos entre ellos.

Unidad 4: Estructuras y Algoritmos Recursivos

OE4.1 Emplear el concepto de recursividad como una alternativa a la estructura de control iterativa.
OE4.2 Aplicar la computación recursiva en la solución de problemas de naturaleza inherentemente autocontenida.
OE4.3 Utilizar árboles binarios de búsqueda para representar grupos de objetos que mantienen entre ellos una relación de orden.
OE4.4 Escribir algoritmos recursivos para manipular estructuras de información recursivas y explicar las ventajas que, en este caso, estos algoritmos tienen sobre los algoritmos iterativos.

Enunciado

La famosa compañía Snakes and Ladders Inc., lo ha contratado para el desarrollo de un programa que permita jugar y también simular el famoso juego Serpientes y Escaleras. El programa puede tener interfaz de texto por consola. El juego debe presentar al usuario una cuadrícula o tabla de ***n*** filas por ***m*** columnas, dentro de la cual hay ***s*** serpientes y ***e*** escaleras.



Cuadrícula de 3 x 8

	~~0~~	~~1~~	~~2~~	~~3~~
~~0~~	17	18	19	20
~~1~~	16	15	14	13
~~2~~	9	10	11	12
~~3~~	8	7	6	5
~~4~~	1	2	3	4

Cada una de las casillas de la cuadrícula puede identificarse a través de un número. La numeración inicia en la casilla inferior izquierda con el número 1, sigue en la casilla inmediatamente a la derecha y así hasta terminar la fila. Luego sube en esa misma columna y se regresa hacia la izquierda, intercalando así la dirección en cada fila, tal como se puede apreciar en la ilustración del típico juego de mesa con serpientes y escaleras pintadas que se encuentra justo arriba a la derecha de este párrafo.

A la izquierda se presenta una cuadrícula con los nombres de las filas, columnas y cada una de las celdas. Tenga en cuenta que la enumeración normal de filas y columnas que hemos usado para las matrices ~~no tiene ningún efecto en esta numeración~~.

El juego consiste en que ***p*** jugadores inician su recorrido a través del tablero en la casilla 1, moviendo por turnos una cantidad de casillas igual a la mostrada por el dado lanzado en ese momento para ese jugador. Los jugadores se mueven en el mismo orden en que están numeradas las casillas. Al final gana el jugador que llegue primero a la casilla numerada con el número mayor.

Si usted utiliza una interfaz por consola, puede representar a cada jugador, cuando esté en una casilla, por cualquiera de los siguientes símbolos: * ! O X % $ # + &. Es importante mantener el mismo símbolo para cada jugador durante toda la partida. Los símbolos pueden ser asignados aleatoriamente al inicio de la partida o pueden ser indicados por el usuario. Si se indican los símbolos, en lugar de dejar un número entero en **p**, allí el usuario ingresaría una cadena sin espacios con los símbolos para cada jugador. De esa manera, la cantidad de símbolos en la cadena indicaría la cantidad de jugadores. Un ejemplo de entrada así sería: **5 4 2 3 #%*** (todo en la misma línea) que indica al programa que cree un tablero de tamaño 5x4, con 2 serpientes, 3 escaleras y 3 jugadores. El primer jugador es #, el segundo es % y el tercero *.

UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
2021-1 - Tarea Integradora 2

Cuadrícula de 3x6 con 5 serpientes

D				**E**	
A		**E**	**B**		**C**
	A	**C**	**D**		**B**

Las serpientes en el juego unen una casilla **con otra** cualquiera **en una fila inferior**. Las serpientes se identifican con letras mayúsculas del alfabeto iniciando en A. En una interfaz por consola, una serpiente puede ser representada a través de la letra que la identifica tanto en la casilla donde inicia como en la casilla donde termina.

A la izquierda podemos ver un ejemplo de escaleras representadas en un tablero.

Las escaleras en el juego unen una casilla **con otra** cualquiera **en una fila superior**. Las escaleras están numeradas desde 1 hasta la e, siendo cada número el identificador de dicha escalera. En una interfaz por consola, una escalera puede ser representada a través del número que la identifica tanto en la casilla donde inicia como en la casilla donde termina.

A la derecha podemos ver un ejemplo de escaleras representadas en un tablero.

Cuadrícula de 3x6 con 4 escaleras

			4		**2**
1	**4**			**3**	
	1		**2**		**3**

El programa debe iniciar con un sencillo menú con 3 opciones. La primera opción es para jugar, la segunda opción es para ver el tablero de posiciones y la tercera opción es salir del programa. Cuando se elige la primera opción, el programa esperará que sean digitados, en la misma línea, 5 números enteros positivos separados por espacio indicando n, m, s, e y p respectivamente.

Cuando el usuario elige jugar, se crea un juego con una cuadrícula de tamaño n**x**m, con s serpientes y e escaleras ubicadas aleatoriamente uniendo cualquiera de las casillas del tablero, con las siguientes restricciones: ninguna escalera inicia en la casilla 1, ninguna serpiente inicia en la casilla n x m, y ninguna casilla de inicio o fin de escalera o serpiente debe coincidir con otro inicio o fin de escalera o serpiente.

Cuando el usuario ingresa los parámetros del juego, el programa le mostrará una cuadrícula formada por corchetes, con las casillas numeradas correctamente y con la ubicación de las escaleras y las serpientes. Por ejemplo, si la cuadrícula es de 4 x 5 se verá como se presenta a la derecha. Las escaleras están en negrita para mejorar la visualización y no confundirlas con los números de las casillas (ponerlas en negrita no es necesario hacerlo en la consola).

```
[203][19 ][18B][17 ][164]
[11C][124][13 ][142][15 ]
[101][ 9A][ 8C][ 73][ 6 ]
[ 1A][ 21][ 3 ][ 4B][ 52]
```

Una vez se despliegue esta visualización de la cuadrícula, el programa queda esperando un salto de línea para iniciar y mostrar el primer tablero. Los tableros del juego irán mostrando la posición de los jugadores en las casillas, pero ya no deben mostrar los números de las casillas, aunque sí las escaleras y las serpientes.

```
[3  ][   ][B  ][   ][4  ]
[C  ][4  ][   ][2  ][   ]
[1  ][A  ][C  ][3  ][   ]
[A$*][1  ][   ][B  ][2  ]
```

En adelante se podrá jugar solamente ingresando un salto de línea, para que juegue el jugador a quien le corresponda el turno. De acuerdo con el tablero anterior, el jugador que inicia es $. Por tanto si se ingresa un salto de línea, entonces dicho jugador lanza el dado (esto lo hace el programa internamente), lo que implica que se genera un número aleatorio entre 1 y 6, se muestra un mensaje indicándolo, por ejemplo: *El jugador $ ha lanzado el dado y obtuvo el puntaje 3*, e inmediatamente se muestra el tablero con la nueva posición.

```
[3  ][   ][B  ][   ][4  ]
[C  ][4  ][   ][2  ][   ]
[1  ][A  ][C  ][3  ][   ]
[A* ][1  ][   ][B$ ][2  ]
```

Tenga en cuenta que si el jugador cae en un inicio de serpiente o de escalera, debe bajar o subir, respectivamente, por dicho elemento del juego y quedar en la casilla correspondiente. Por ejemplo, si el jugador * que juega a continuación, ahora obtiene un valor de 4 en el dado, se moverá hasta la casilla 5 en la cual empieza la escalera 2 y subirá por ella hasta la casilla correspondiente que es la 14 (ver el primer tablero mostrado por el juego por si no está clara la numeración de las casillas).

```
[3  ][   ][B  ][   ][4  ]
[C  ][4  ][   ][2* ][   ]
[1  ][A  ][C  ][3  ][   ]
[A  ][1  ][   ][B$ ][2  ]
```

Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **num** y luego se da salto de línea, el programa mostrará la misma cuadrícula que muestra al inicio, con las casillas numeradas, las serpientes y las escaleras. Esperará entonces un salto de línea para mostrar el tablero actual (con las serpientes, escaleras y posición actual de los jugadores) y continuar con el juego.

Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **simul** y luego se da salto de línea, el programa empezará en modo simulación, que consiste en ir mostrando lo que cada jugador a su turno juega, con el tablero correspondiente de cada nueva posición, esperando 2 segundos entre cada jugada, pero sin esperar ningún salgo de línea.

Si en lugar de simplemente ingresar un salto de línea al programa, se escribe la palabra **menu** y luego se da salto de línea, el juego se corta sin terminar y el programa regresa al menú principal, mostrándose sus opciones.

El juego termina cuando uno de los jugadores llegue a la última casilla (la casilla cuya numeración es n **x** m). En ese caso debe decir: El jugador **Z** ha ganado el juego, con **Y** movimientos. Donde **Z** es el símbolo del jugador ganador y **Y** es la cantidad de veces que el jugador lanzó el dado en ese juego. A continuación de este mensaje, se pide el nombre o nickname del jugador ganador y posteriormente se muestra el menú principal del programa.

Si se regresa al menú principal cuando algún jugador gana el juego, se calcula un puntaje para el usuario que es igual a la cantidad de movimientos multiplicado por la cantidad total de casillas del tablero. Este puntaje debe ser almacenado en un árbol binario de búsqueda ordenado inversamente por puntaje. La opción 2 mostrará un listado de los nombres o nicknames de los jugadores, sus símbolos y sus respectivos puntajes, resultado de recorrer el árbol binario de búsqueda en inorden. Cada vez que el programa espere solo un salto de línea para continuar, informe al usuario con un mensaje**.**

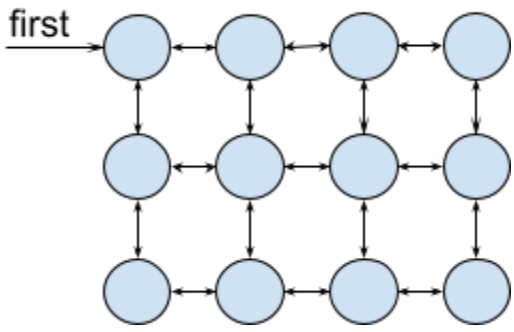
Condiciones:

i. La cuadrícula debe ser modelada e implementada utilizando listas enlazadas. Puede ser una lista enlazada de listas enlazadas.

ii. No es posible utilizar ningún arreglo (de ninguna dimensión) ni arraylist (ni ninguna colección de Java) en este programa. La **única excepción** es el arreglo que devuelve el split que se le hace al String que contiene los valores m, n, s, e y p.

iii. No es posible utilizar ciclos en este programa. Todas las repeticiones deben hacerse utilizando **recursión**.

Se sugiere que su lista enlazada de listas enlazadas tenga el siguiente enlazamiento:



UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
2021-1 - Tarea Integradora 2

Usted debe entregar los siguientes artefactos y condiciones:

1. **[10%]** Especificación de Requerimientos Funcionales.
2. **[25%]** Diseño completo del diagrama de clases, incluyendo el paquete del modelo y la interfaz con el usuario.
3. **[50%]** Implementación completa y correcta del modelo, y la ui.
4. **[15%]** Usted debe entregar el enlace del repositorio en GitHub o GitLab con los elementos anteriores. El nombre del repositorio debe estar en inglés, en minúsculas y si tiene varias palabras, éstas van separadas por un guión. Su repositorio debe corresponder con un proyecto de eclipse o su IDE favorito. Debe tener al menos 10 commits con diferencia de 1 hora entre cada uno de ellos. En el repositorio o proyecto de eclipse debe haber un directorio llamado **docs/** en el cual deberán ir cada uno de los documentos del diseño.

   El **readme.md** del repositorio debe explicar brevemente (en inglés) de qué se trata el proyecto. Es importante que ni en el nombre, ni la explicación del readme hagan referencia a una "tarea" de un curso ni nada parecido, sino que explique de forma independiente al curso, de qué se trata el problema y la solución del mismo. Deben enlazar los archivos que documentan el proyecto (en formato pdf) y deben especificar las condiciones técnicas del mismo (lenguaje, sistema operativo, ambiente de desarrollo e instalación). **El último commit en la rama master debe estar marcado con un tag llamado Milestone1**. La fecha del commit marcado con este tag debe ser previo a la fecha y hora límite de la entrega. Usted podrá seguir haciendo commits posteriores a la fecha de entrega, pero la revisión se hará sobre la versión del proyecto marcada con esta etiqueta.

   Recuperación:
   Después de la publicación de la calificación de la primera entrega, su equipo tendrá una semana, para hacer ajustes y completar los elementos del proyecto que desee si desea que sea evaluado nuevamente. Esa segunda entrega deberá ser etiquetada como **Milestone2**.
   En esta versión recuperatoria del proyecto, al **readme.md** del repositorio debe agregársele una sección que indique concretamente qué elementos fueron agregados o modificados en esta versión (para que quien califique conozca qué es lo nuevo que debe revisar). El título de la sección puede ser **changelog**. También está la opción de hacer esta descripción de cambios de la nueva versión en un archivo aparte, **enlazado en el readme**. Así lo haga en un archivo aparte o dentro del mismo readme, por favor siga las recomendaciones en https://keepachangelog.com.

Los requerimientos funcionales y el diagrama de clases (en imagen de alta calidad) deben entregarse en un mismo archivo en formato **pdf**, bien organizado por secciones y títulos, con hoja de portada.

Importante: su repositorio debe ser privado hasta la hora y fecha máxima de entrega, después de la cual usted debe hacerlo público para que pueda ser revisado.

Nota: Esta tarea se evaluará con la rúbrica de la Tarea Integradora 2 Snakes And Ladders. Se recomienda revisar la rúbrica con la que será evaluada su entrega.

Fecha Máxima de Entrega: 30 de Abril de 2021. 23:55. La tarea se califica sobre 5.0.

Bonificación por Pronta Entrega: 26 de Abril de 2021. 23:55. La tarea se califica sobre **6.0**.

VIDEO EXPLICATIVO DE UNA MATRIZ DOBLEMENTE ENLAZADA BASE - PARTE 1

VIDEO EXPLICATIVO DE UNA MATRIZ DOBLEMENTE ENLAZADA BASE - PARTE 2

El código explicado en este video puede ser utilizado por ustedes en el desarrollo de su proyecto. El código funciona, de acuerdo como se muestra en el video, pero entrega sin ninguna garantía. Usted debe dar los créditos al autor en caso de utilizarlo en su propio desarrollo.