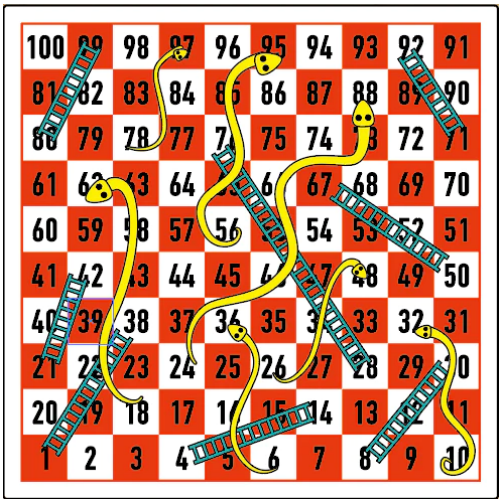
UNIVERSIDAD ICESI

Algoritmos y Programación II
Facultad de ingeniería

TAREA INTEGRADORA 1

LISTAS ENLAZADAS, ÁRBOLES, EXCEPCIONES

SNAKES AND LADDERS



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 Escaleras y Serpientes. 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.

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.

Cuadrícula de 3x6

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

Descripción del tablero de juego
El juego consiste en ***3 jugadores*** que inician su recorrido a través del tablero. Todos parten de la casilla 1, moviéndose por turnos a lo largo de las casillas. Gana el jugador que llegue primero a la última casilla.

Puede representar a cada jugador con cualquiera de los siguientes símbolos: * ! O X % $ # + &.

Las serpientes en el juego **unen una casilla con otra** cualquiera **en una casilla inferior**. Las serpientes se identifican con letras mayúsculas del alfabeto iniciando en A. A continuación podemos ver un ejemplo de escaleras representadas en un tablero 3x6.

Cuadrícula de 3x6 con 3 serpientes

A			B		C
	A	C			B

Las escaleras en el juego unen una casilla **con otra** cualquiera **en una casilla 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.

Cuadrícula de 3x6 con 3 escaleras

	1			3	
		3		2	
			1	2	

El programa debe iniciar con un sencillo menú con 2 opciones. La primera opción es para jugar y la segunda opción es salir del programa.

```
1. Jugar
2. Salir
```

Cuando se elige la primera opción, el programa pedirá cuál es el número de **filas** y **columnas** del tablero de juego. El programa también pedirá el número de escaleras y serpientes que tendrá el juego.

Entonces se crea un juego con una cuadrícula de tamaño **nxm**, 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.

Desarrollo del juego
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. En la cuadrícula 3x6 del ejemplo se verá como se presenta a continuación.

```
[13]   [14]   [15]   [16]   [17]   [18]
[12]   [11]   [10]   [9]    [8]    [7]
[1*$%] [2]    [3]    [4]    [5]    [6]
```

Donde *, $ y % son los jugadores ubicados en las casillas

A partir de aquí inicia el juego. Debe definir un sistema de turnos. En su turno cada jugador tiene el siguiente menú

```
Jugador $, es tu turno
1. Tirar dado
2. Ver escaleras y serpientes
```

Tirar Dado

Aleatoriamente el jugador lanza el dado y N casillas hacia adelante. Si cae en una serpiente, el jugador se retrasa a lo largo del tablero. Si cae en una escalera, el jugador se adelanta. Si cae en una casilla normal, el jugador queda en esa casilla

Ver escaleras y serpientes

Con esta opción el jugador puede ver las escaleras y serpientes del tablero

```
[ ]  [1]  [ ]   [ ]  [3]  [ ]
[A]  [ ]  [3]   [B]  [2]  [C]
[ ]  [A]  [C]   [1]  [2]  [B]
```

Por ejemplo si el primer jugador (*) lanza el dado y saca 5, el programa muestra inmediatamente el tablero con la nueva posición del jugador

```
[13]   [14]   [15]   [16]   [17]   [18]
[12]   [11]   [10]   [9]    [8]    [7]
[1$%]  [2]    [3]    [4]    [5]    [6*]
```

Note que el jugador cae en la salida de la serpiente, por lo tanto no debe atravesarla

Es turno del segundo jugador (%). Usa la opción de ver las escaleras y serpientes

```
[ ]  [1]  [ ]   [ ]  [3]  [ ]
[A]  [ ]  [3]   [B]  [2]  [C]
[ ]  [A]  [C]   [1]  [2]  [B]
```

Luego el jugador (%) lanza el dado y saca un 3, el programa muestra

```
[13]   [14%]  [15]   [16]   [17]   [18]
[12]   [11]   [10]   [9]    [8]    [7]
[1$]   [2]    [3]    [4]    [5]    [6*]
```

Note que el jugador (%) atravesó la escalera (1)

Es turno del jugador ($), tira el dado y saca un 6, el programa muestra

```
[13]   [14]   [15]   [16]   [17]   [18]
[12]   [11]   [10]   [9%]   [8]    [7]
[1]    [2]    [3$]   [4]    [5]    [6*]
```

Note que el jugador ($) atravesó la serpiente (C)

Final del juego

El juego finaliza cuando un jugador llega a la última casilla. En ese momento el jugador obtiene un puntaje que puede calcularse de la siguiente fórmula:

```
Puntaje = (600 - t) / 6
```

Donde t es el tiempo en segundos transcurrido entre el inicio del juego y cuando uno de los jugadores llega a la meta. Esta fórmula está hecha de modo que a partir de los 10 minutos, el puntaje comienza a ser negativo.

Puntaje final

Luego de que el jugador llega a la meta, se muestra una lista ordenada del puntaje más alto al puntaje más bajo. Debe ser almacenado en un árbol binario de búsqueda.

NOTA: Como NO se debe implementar persistencia, los datos del registro de puntajes se eliminan en cuanto cierre el programa.

ACLARACIÓN
El juego termina cuando uno de los jugadores llega a la meta. Sólo a ese jugador se le calcula su puntaje. Para que hayan más jugadores en el scoreboard, se deberá jugar nuevamente sin cerrar el programa.

Esto quiere decir que al finalizar una partida, el programa debe volver a comenzar nuevamente.

Condiciones:

1. La cuadrícula debe ser modelada e implementada utilizando listas enlazadas.
2. No es posible utilizar ningún arreglo, ni arraylist, ni ninguna colección de Java en este programa.
3. No es posible utilizar ciclos en este programa. Todas las repeticiones deben hacerse utilizando **recursión**.
4. Todas las escaleras y serpientes deben ser modeladas como conexiones entre nodos de la estructura enlazada

Entregas

1. **Requerimientos (15%)**
   Especifique los requerimientos a partir de lo que aparece en el enunciado de la tarea integradora.
   Entrega: final de semana 3

2. **Diseño preliminar**
   Realice un diagrama de clases UML con el diseño inicial del programa. Este plano debe comunicar lo que tiene pensado hacer para modelar el programa de la tarea integradora. Esta entrega preliminar no tiene nota.
   Entrega: Final de semana 4

3. **Implementación (70%) y diseño UML (15%) completos**
   Esta entrega debe tener la implementación funcionando perfectamente acorde a los requerimientos.
   Anexe el diseño, esta vez en su versión final. Este diseño tendrá nota a diferencia del diseño preliminar
   Entrega: Final de semana 6