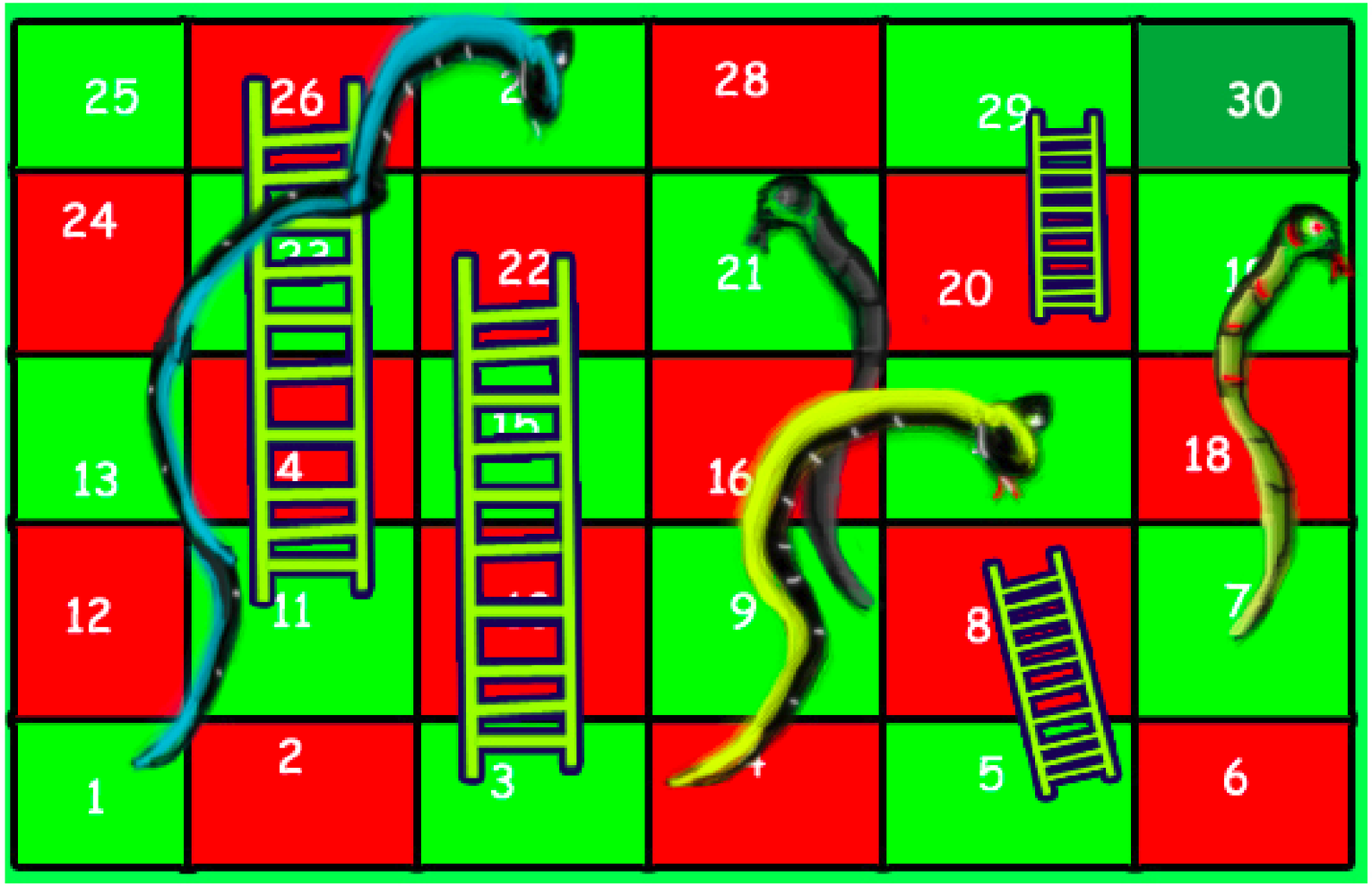
Snake and Ladder

By: Srujan Vasudevrao Deshpande and Vaibhav Gupta

Introduction

- Problem Statement: Implementation of Snake and Ladder Game using Graphs and Multilist.
- We are going to implement a 5x6 snake and ladder game. (30 Squares)
- Rules: You start at 1, end at 30, snake or ladder require 0 moves.

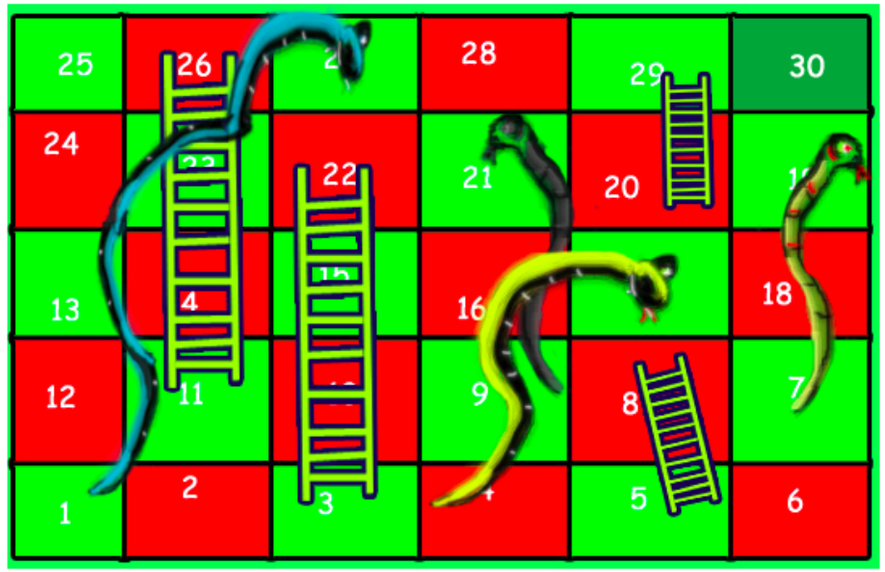
The Game Board



Demonstration

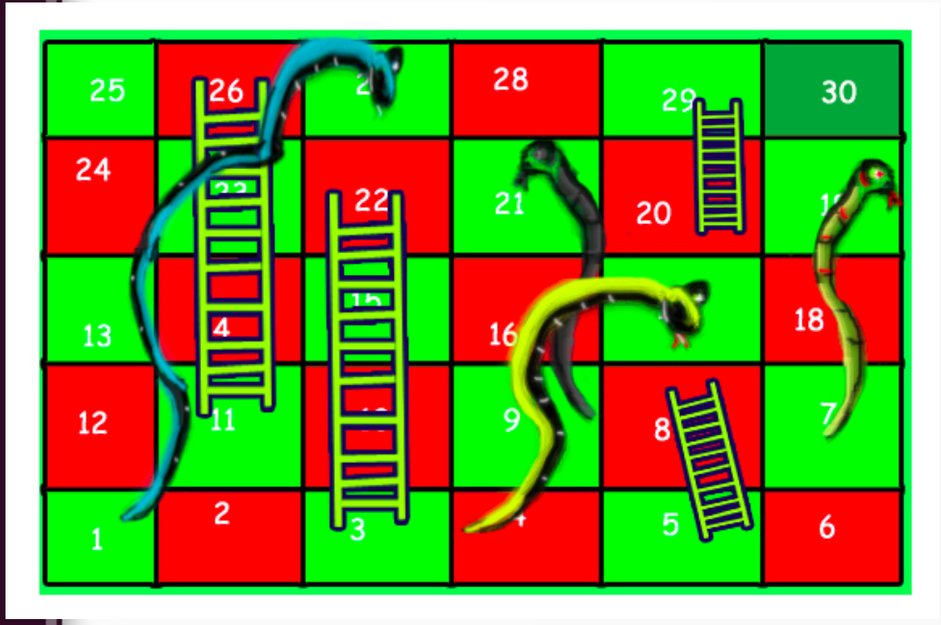
srujan@SVD-UPC: ~/Projects/snake-ladder

```
Roll:1, Fast-Forward:2, Restart: 8, Quit:9
+-----------------+
|25|26|27|28|29|30|
|24|23|22|21|20|19|
|13|14|15|16|17|18|
|12|11|10|09|08|07|
|01|02|03|04|05|06|
+-----------------+
0 Moves - Player 1 is on: 1
0 Moves - Player 2 is on: 1
~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~
Player 1's turn
```



Demonstration

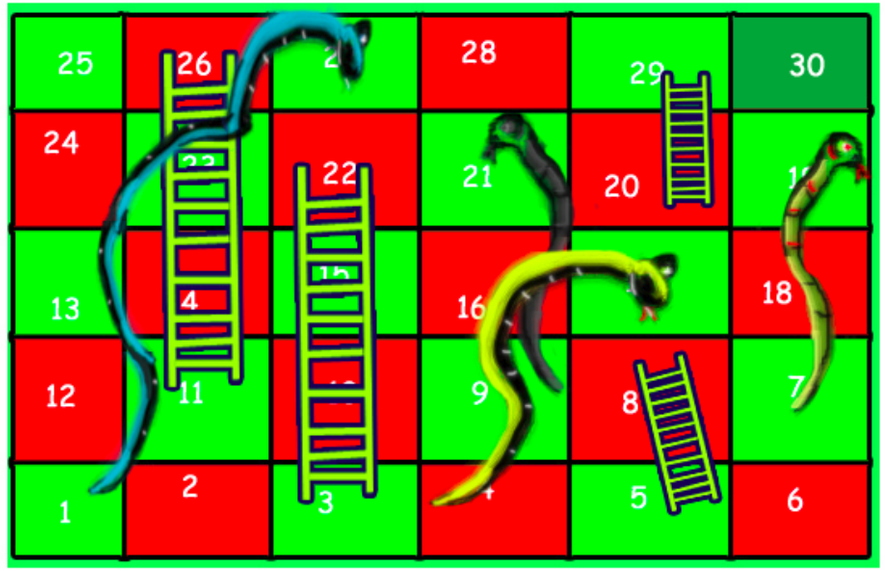
```
srujan@SVD-UPC: ~/Projects/snake-ladder
Roll:1, Fast-Forward:2, Restart: 8, Quit:9
+-----------------+
|25|26|27|28|29|30|
|24|23|22|21|20|19|
|13|14|15|16|17|18|
|12|11|10|09|08|07|
|01|02|03|04|05|06|
+-----------------+
1 Moves - Player 1 is on: 8
0 Moves - Player 2 is on: 1
~~~~~~~~~~~~~~~~~~~
Player 1 rolled a 4
~~~~~~~~~~~~~~~~~~~
Player 2's turn
1
```



Demonstration

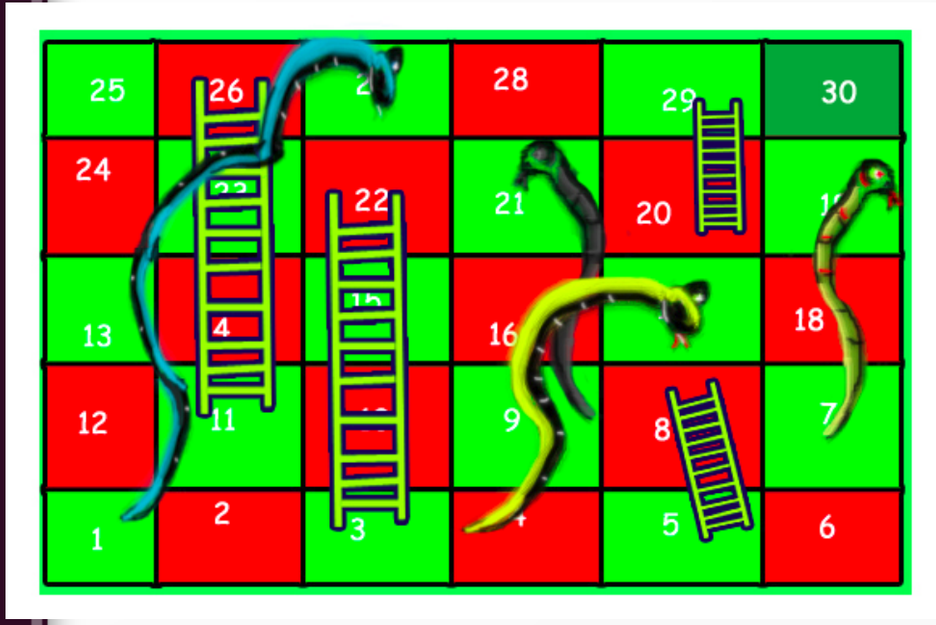
srujan@SVD-UPC: ~/Projects/snake-ladder

```
Roll:1, Fast-Forward:2, Restart: 8, Quit:9
+-----------------+
|25|26|27|28|29|30|
|24|23|22|21|20|19|
|13|14|15|16|17|18|
|12|11|10|09|08|07|
|01|02|03|04|05|06|
+-----------------+
1 Moves - Player 1 is on: 8
1 Moves - Player 2 is on: 7
~~~~~~~~~~~~~~~~~~~
Player 2 rolled a 6
~~~~~~~~~~~~~~~~~~~
Player 1's turn
2
```



Demonstration

```
srujan@SVD-UPC: ~/Projects/snake-ladder
Roll:1, Fast-Forward:2, Restart: 8, Quit:9
+-----------------+
|25|26|27|28|29|30|
|24|23|22|21|20|19|
|13|14|15|16|17|18|
|12|11|10|09|08|07|
|01|02|03|04|05|06|
+-----------------+
6 Moves - Player 1 is on: 28
5 Moves - Player 2 is on: 29
~~~~~~~~~~~~~~~~~~~
Player 1 rolled a 5
~~~~~~~~~~~~~~~~~~~
Player 2's turn
You rolled a 1
Player 2 wins!
srujan@SVD-UPC:~/Projects/snake-ladder$
```



Approach

- The multilist stores the current node, next node, the link of the current node and the link of the next node.
- Each box is a node
- Each edge is a list
- The multilist is stored in a csv file. The program reads it and then generates the actual multilist.
- In order to play using fast-forward mode, we skip asking the user to input and directly roll the dice.

Graphs

- A graph is another word for a network, i.e., a set of objects (called vertices or nodes) that are connected together. The connections between the vertices are called edges or links.

Algorithms – Dice Roll

- C does not have true random number generation and can only generate pseudo random numbers. This means every time the program re-runs, the random numbers repeat.
- We set the current time as the seed for the random number at the start of every game.
- Thus, every game will have different random numbers and different dice rolls.

Algorithms - Moves

- The number rolled by the dice is passed to the move function which first checks if there are that many spaces left to move.
- Then it moves n-1 spaces.
- For the last move, it checks if there is a snake or ladder which can be taken. If yes, it takes that, else it moves normally forward.
- At the end we check if the last square had been reached.

Any Questions?