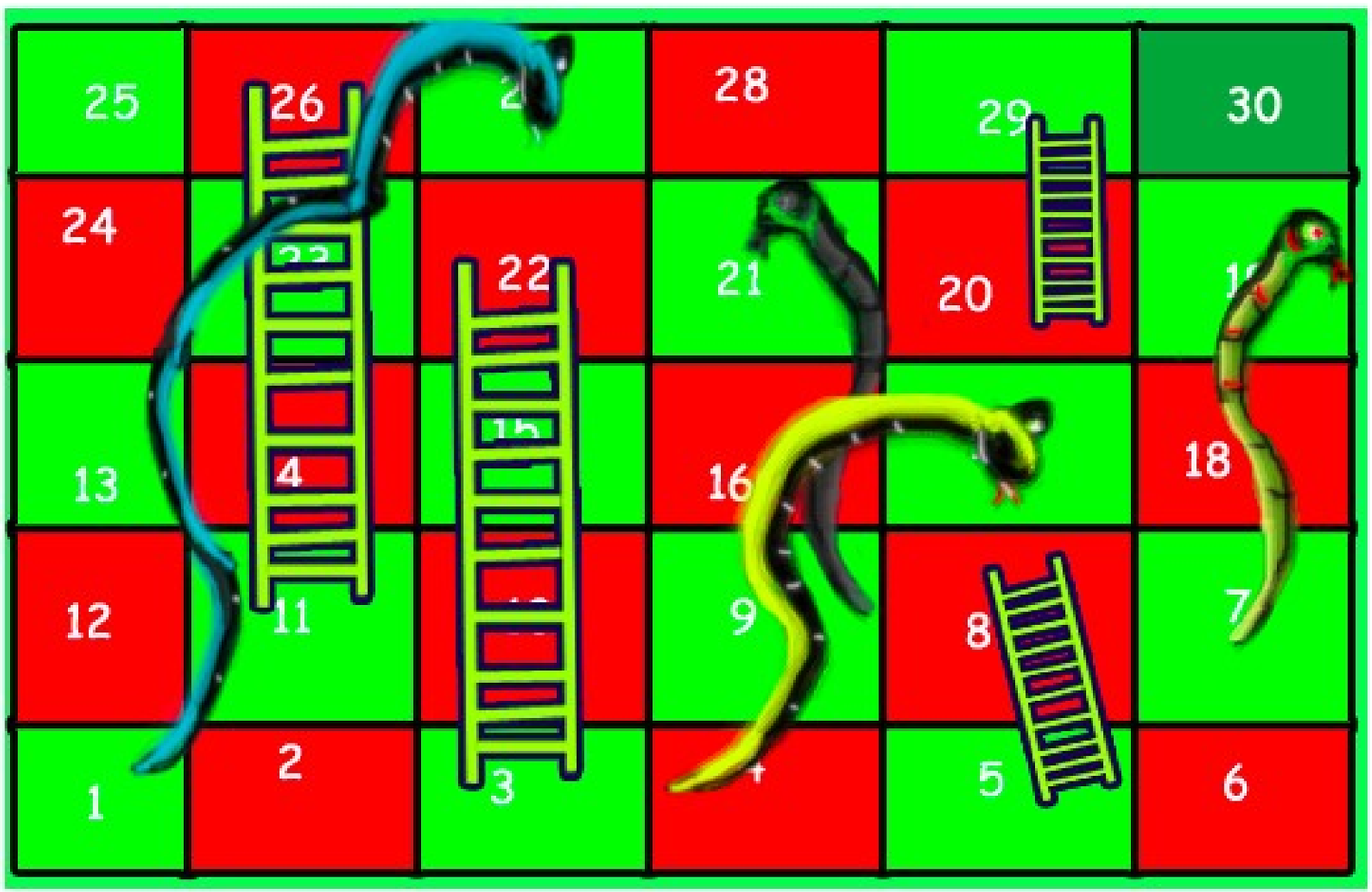
Snake and Ladder

By: Srujan Vasudevrao Deshpande and Vaibhav Gupta

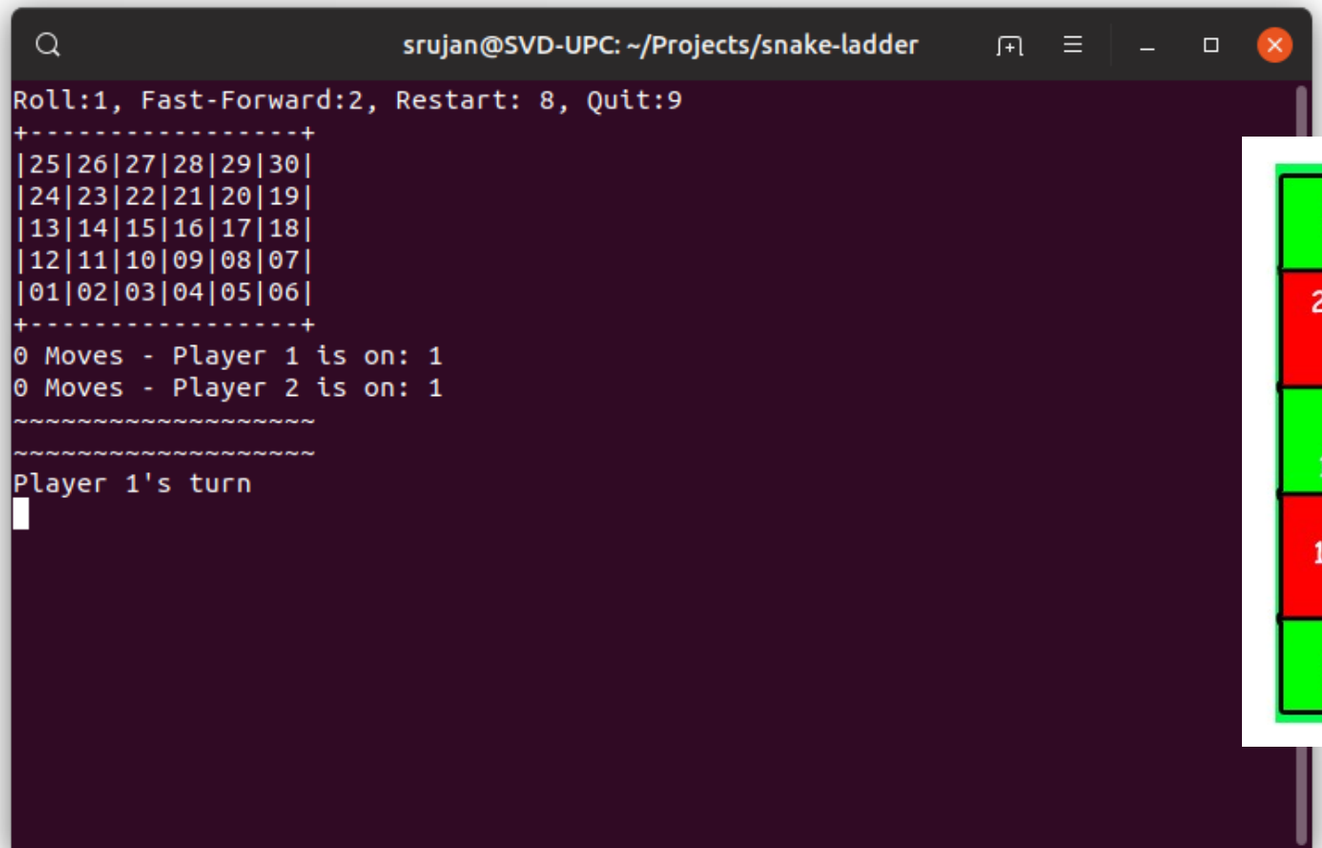
Introduction

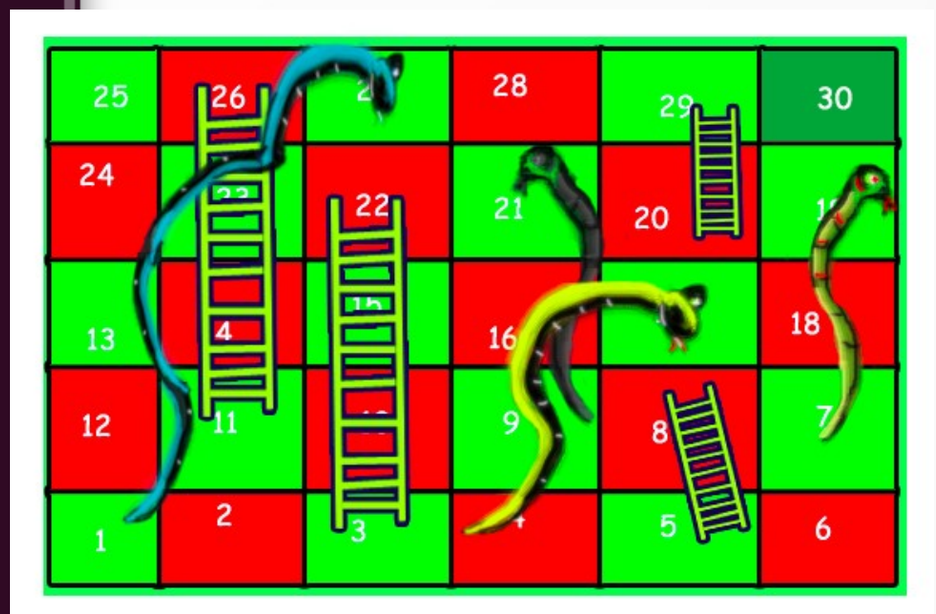
- Problem Statement: Implementation of Snake and Ladder Game using Graphs and Multilist.
- We are going to implement a 5x6 snake and ladder game. (30 Squares)
- Rules: You start at 1, end at 30, snake or ladder require 0 moves.

The Game Board

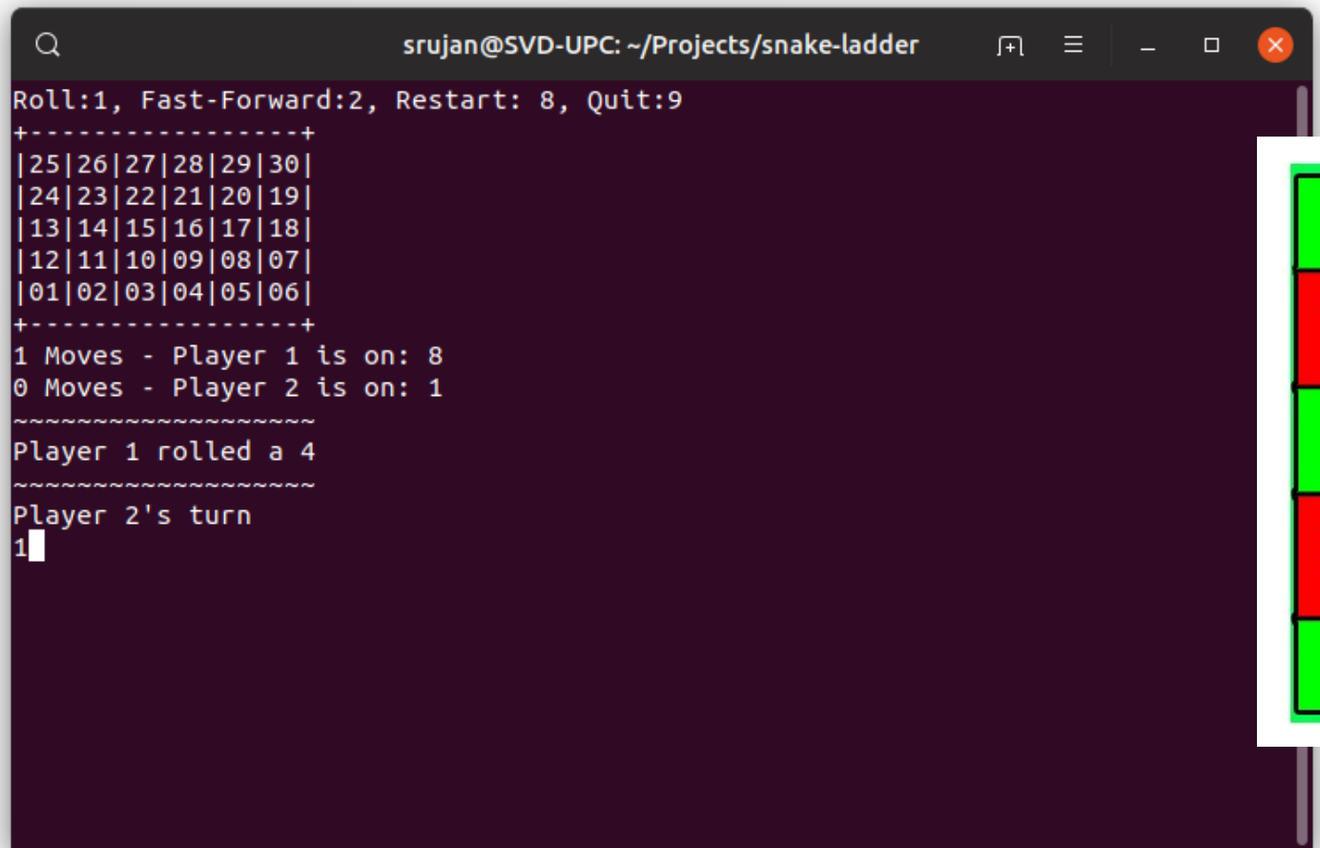


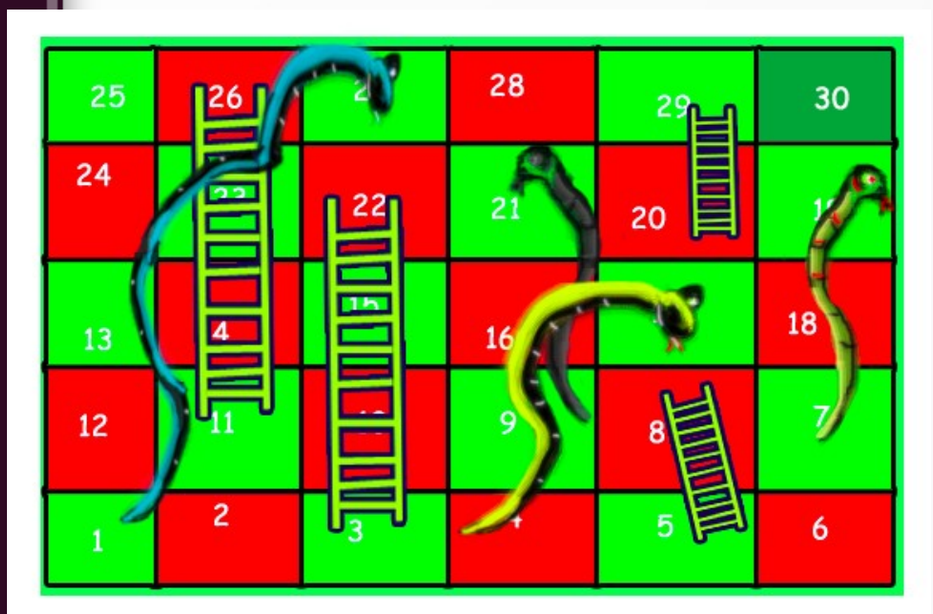
Demonstration



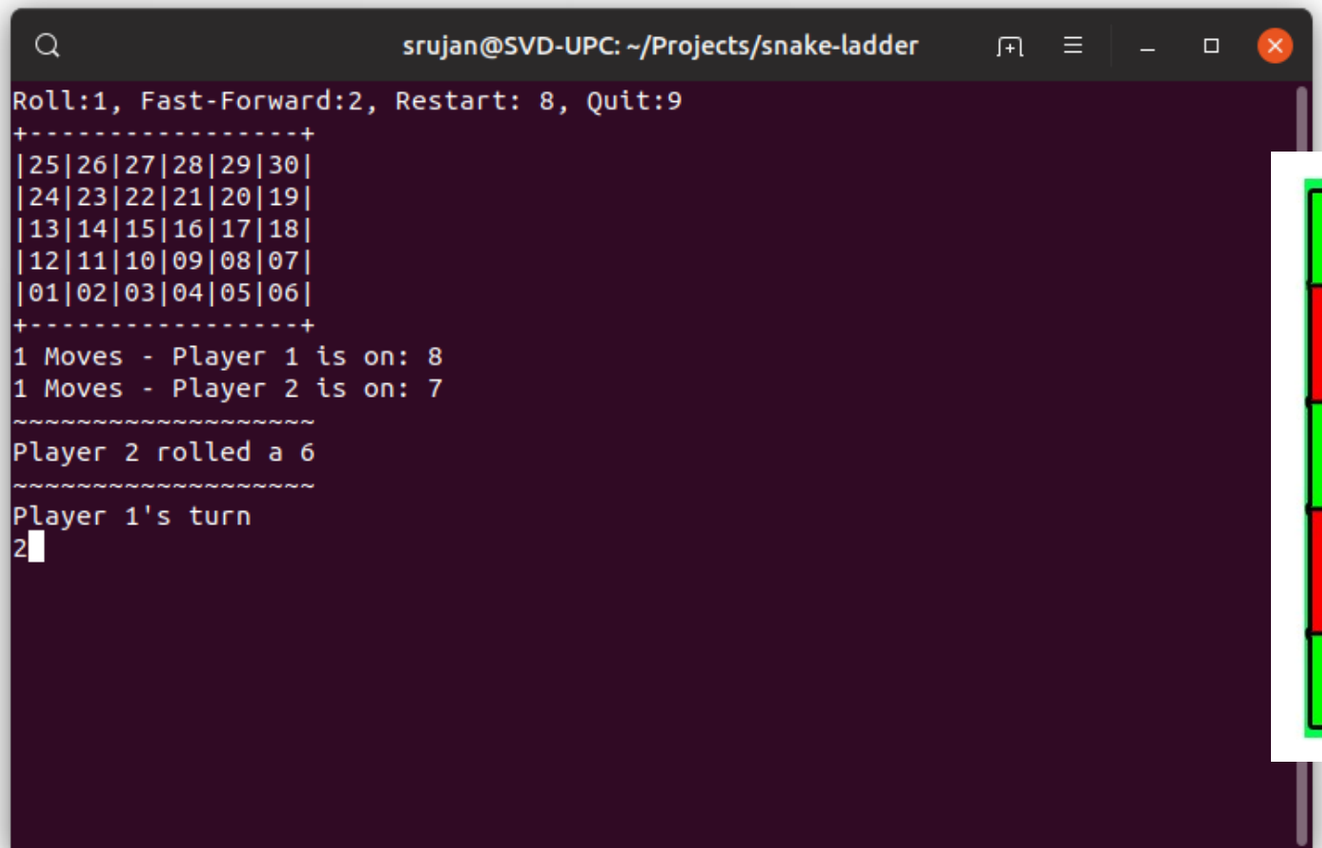


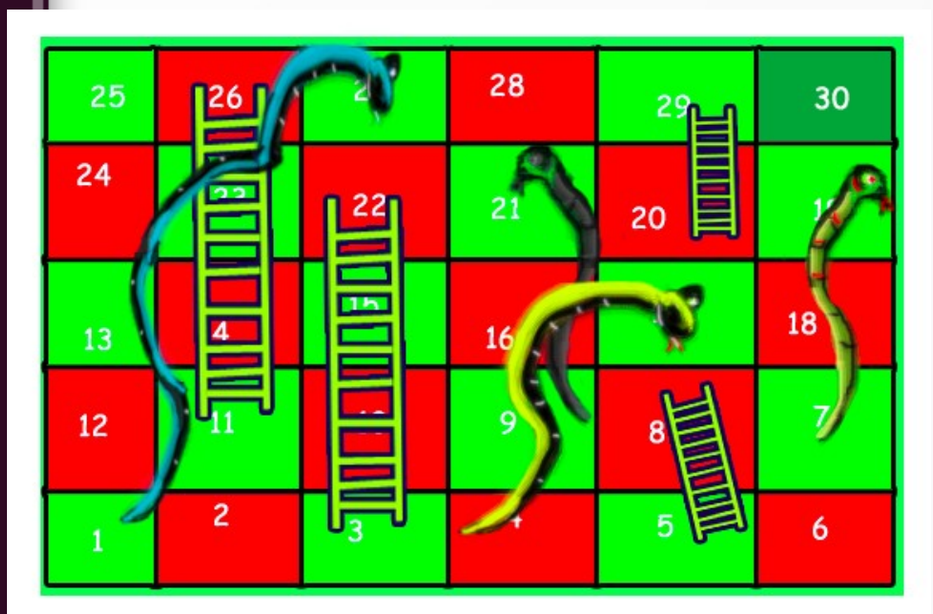
Demonstration



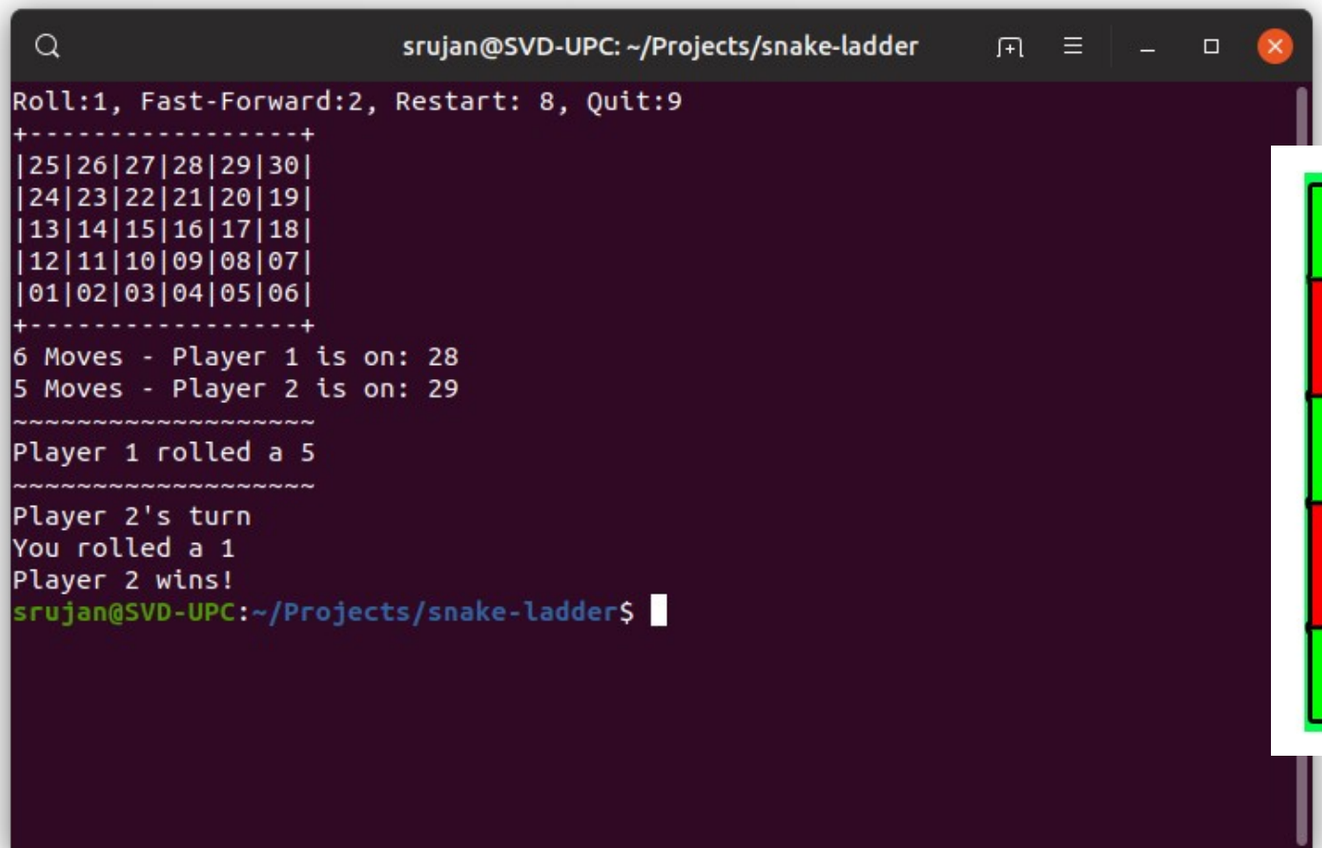


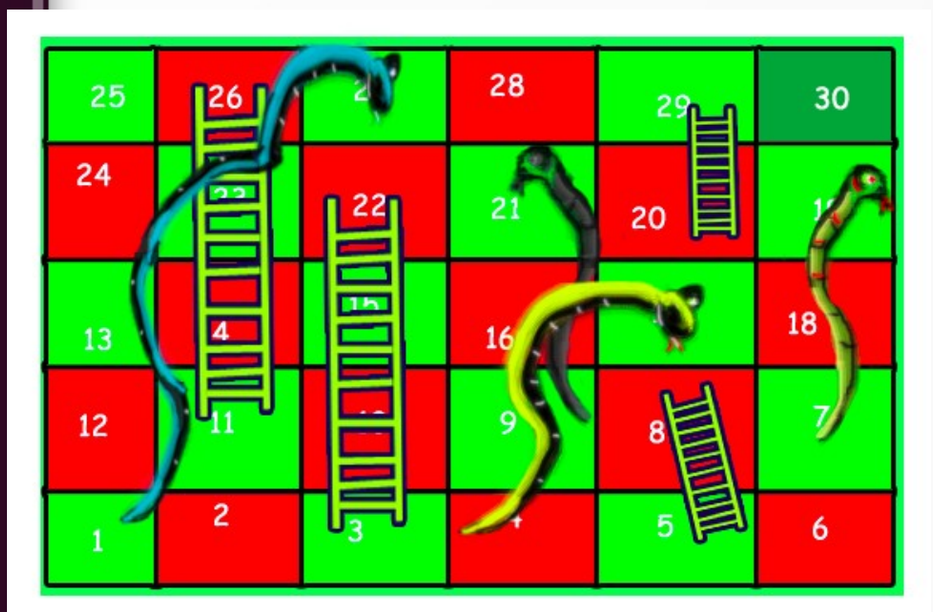
Demonstration





Demonstration





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.

ADT Definition

Graph data structre implemented using multilist

- List** multilist is an array of List structure pointers
- Each List array contains 2 nodes and 2 links
- Int createMultiList() function creates the multilist and initializes the values in it

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?