SRI VENKATESWARA COLLEGE OF ENGINEERING
(AUTONOMOUS)
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu Accredited by NBA, New Delhi & NAAC with 'A' grade)
Karakambadi Road, TIRUPATI – 517507
2023 – 2026

Dr. V. Lakshmi Devi Ph.D.
Head of the Department

- **Name of the Trainee** :Shaik Mulla Baba Fakruddin
- **Name of the Company** :ByteXL TechED Pvt. Ltd
- **Name of the Supervisor/Guide** :M.Pranay
- **Title of Report** :Snake And Ladder Using C
- **Field of Training** : Programming Technologies
- **Area of the project** : C Programming

Abstract

Snakes and Ladders is one of those board games that have remained a very popular classic and originated in ancient India, known as Moksha Patam. It was later commercialized in the Western world. Traditionally, this game is played by two or more players on a grid-based board numbered from 1 to 100. The basic objective of the game is to travel through a grid-shaped board by rolling a die to land on a place that has your corresponding number on it. The board contains "ladders," which help the players to advance quickly by climbing to a higher-numbered square, and "snakes," which send players back to a lower-numbered square, representing setbacks. It is a race to the final square first, blending luck, strategy, and life lessons about perseverance and chance. Over time, Snakes and Ladders has evolved into different versions, including digital adaptations, while retaining its core mechanics and universal appeal. It remains a popular tool for teaching children numbers, counting, and moral values, as well as a source of entertainment for all ages.

Introduction :

Snake And Ladder

Snakes and Ladders is a timeless board game with origins in ancient India, where it was known as Moksha Patam. Originally designed as a moral and spiritual teaching tool, the game symbolized the journey of life—ladders represented virtues leading to enlightenment, while snakes signified vices causing setbacks. Over time, it evolved into a widely enjoyed family game, transcending its religious roots.

The traditional board consists of a 100-square grid, where players start at square 1 and aim to reach square 100 by rolling a die. Ladders help players advance quickly, while snakes send them backward, creating an unpredictable and engaging experience. This combination of chance and strategy makes the game exciting for players of all ages.

One of the game's strengths is its simplicity. With minimal rules, it is accessible to both children and adults, serving as an educational tool for learning counting, numbers, and perseverance. It subtly teaches life lessons—just as in life, success and failure often depend on factors beyond our control, but persistence is key to reaching one's goal.

Snakes and Ladders was introduced to England in the 19th century, where it was adapted for children and commercialized. Its popularity spread across Europe and North America, and in recent years, digital versions have made it even more accessible. Despite technological advancements, the core gameplay remains unchanged, preserving its classic appeal.

The game has inspired numerous adaptations, from educational versions to themed boards featuring popular characters. Some variations replace snakes and ladders with elements like chutes and slides or rockets and black holes, ensuring its relevance in different contexts.

Beyond entertainment, Snakes and Ladders has also been used in psychological studies to examine decision-making and risk-taking. Researchers analyze how individuals react to success and failure, making it a valuable tool for behavioral studies.

In conclusion, Snakes and Ladders is more than just a game; it is a cultural artifact that has endured through centuries. Its origins in moral teachings have given way to a beloved family pastime, yet its essence remains the same. Whether played on a physical board or a digital screen, its blend of luck, strategy, and life lessons continues to captivate players worldwide, securing its place as a timeless classic.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

Hardware - intel i3
Speed - 2.1 GHz
RAM - 4GB
Hard Disk - 100 GB SSD
Floppy Drive - 2.88 MB
Key Board - Standard Windows Keyboard
Mouse - Two or Three-Button Mouse

SOFTWARE REQUIREMENTS:

Operating System: Windows 10, 11
Technology: C
Compiler: byteXL TechEd editor(https://bytexl.app/editor).

Code

```
// C Program to implement Snake and Ladder Game
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// Function to roll a six-sided die
int rollDie() { return rand() % 6 + 1; }
// global variables to store postions of player1 and player2
int player1 = 0, player2 = 0;
// Function to print the board
void printBoard()
{
	// logic to print a snake-ladder Game board
	// programmer can implement their own logic for the board,
	// this is one way to print a snake ladder board.
	int board[101];
	for (int i = 1; i <= 100; i++) {
		board[i] = i;
	}
	int alt = 0; // to switch between the alternate nature of the board
	int iterLR = 101; // iterator to print from left to right
	int iterRL = 80; // iterator to print from right to left
	int val = 100;
	while (val--) {
		if (alt == 0) {
			iterLR--;
			if (iterLR == player1) {
				printf("#P1 ");
			}
			else if (iterLR == player2) {
				printf("#P2 ");
			}
			else
				printf("%d ", board[iterLR]);
			if (iterLR % 10 == 1) {
				printf("\n\n");
				alt = 1;
				iterLR -= 10;
			}
		}
		else {
			iterRL++;
			if (iterRL == player1) {
				printf("#P1 ");
			}

			else if (iterRL == player2) {
				printf("#P2 ");
			}
			else
			    printf("%d ", board[iterRL]);
			if (iterRL % 10 == 0) {
				printf("\n\n");
				alt = 0;
```

```
                                        iterRL -= 30;
                                }
                        }
                        if (iterRL == 10)
                                break;
                }
                printf("\n");
}
// Function to move the player
int movePlayer(int currentPlayer, int roll)
{
                int newPosition = currentPlayer + roll;
                // Define the positions of snakes and ladders on the
                // board
                int snakesAndLadders[101];
                for (int i = 0; i <= 100; i++) {
                        snakesAndLadders[i] = 0;
                }
                // here positive weights represent a ladder
                // and negative weights represent a snake.
                snakesAndLadders[6] = 40;
                snakesAndLadders[23] = -10;
                snakesAndLadders[45] = -7;
                snakesAndLadders[61] = -18;
                snakesAndLadders[65] = -8;
                snakesAndLadders[77] = 5;
                snakesAndLadders[98] = -10;
                int newSquare
                        = newPosition + snakesAndLadders[newPosition];
                if (newSquare > 100) {
                return currentPlayer; // Player cannot move beyond // square 100
                }
                return newSquare;
}
int main()
{
                srand(time(0)); // Initialize random seed
                int currentPlayer = 1;
                int won = 0;
                printf("Snake and Ladder Game\n");
                while (!won) {
                        printf(
                                "\nPlayer %d, press Enter to roll the die...",
                                currentPlayer);
                        getchar(); // Wait for the player to press Enter
                        int roll = rollDie();
                        printf("You rolled a %d.\n", roll);
                        if (currentPlayer == 1) {
                                player1 = movePlayer(player1, roll);
                                printf("Player 1 is now at square %d.\n\n",
                                        player1);
                                printBoard();
                                if (player1 == 100) {
                                        printf("Player 1 wins!\n");
                                        won = 1;
                                }
```

```
                }
                else {
                        player2 = movePlayer(player2, roll);
                        printf("Player 2 is now at square %d.\n\n",
                                player2);
                        printBoard();
                        if (player2 == 100) {
                                printf("Player 2 wins!");
                                won = 1;
                        }
                }
                // Switch to the other player
                currentPlayer = (currentPlayer == 1) ? 2 : 1;
        }
        return 0;
}
```

Code Explanation

Key Features:

Rolling the Die: The rollDie function will return a number between 1 and 6, simulating a six-sided die.

Movement of the Player: The movePlayer function determines a new position based on the current position of the player and the roll of the die. It also accounts for "snakes" and "ladders" by having the function compare the new position of the player to a set list of positions for snakes and ladders.

Board Printing: The printBoard function prints the 10x10 board in a zigzag pattern. Players' positions are indicated by #P1 for Player 1 and #P2 for Player 2 so that players know where they are on the board.

Game Loop: The main function runs Player 1 followed by Player 2 in an alternating manner. After every roll, the board is printed, and the player's position is updated. This continues until a player reaches the finish line (position 100), and at that point, the game stops and declares a winner.

Main Code Segments:

Snakes and Ladders Definition: The snakes and ladders positions are hardcoded in an array snakesAndLadders[101], where positive numbers represent ladders and negative numbers represent snakes. For example:

A ladder at position 6 moves the player to position 40.
A snake at position 23 sends the player back to position 13.
Main Game Loop: The main game loop asks the players to press Enter to roll the die. The game then updates the positions and checks for a winner after each roll.

Improvements or Changes:

Adding More Snakes and Ladders: You can add more snakes or ladders to the snakesAndLadders array to make the game more interesting.
Handling Edge Cases: Consider adding more checks or messages to handle situations where the player's roll would move them past 100, ensuring they stay within bounds.

Example Output:

css
Copy
Snake and Ladder Game

Player 1, press Enter to roll the die.
You rolled a 3.
Player 1 is now at square 3.

1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

.

Player 1 wins!
The program alternates between the two players and displays their current position after each roll, along with the updated board.

```
Player 1, press Enter to roll the die...You rolled a 2.
Player 1 is now at square 2.

100 99 98 97 96 95 94 93 92 91

81 82 83 84 85 86 87 88 89 90

80 79 78 77 76 75 74 73 72 71

61 62 63 64 65 66 67 68 69 70

60 59 58 57 56 55 54 53 52 51

41 42 43 44 45 46 47 48 49 50

40 39 38 37 36 35 34 33 32 31

21 22 23 24 25 26 27 28 29 30

20 19 18 17 16 15 14 13 12 11

1 #P1 3 4 5 6 7 8 9 10


Player 2, press Enter to roll the die...You rolled a 2.
Player 2 is now at square 2.

100 99 98 97 96 95 94 93 92 91

81 82 83 84 85 86 87 88 89 90

80 79 78 77 76 75 74 73 72 71

61 62 63 64 65 66 67 68 69 70

60 59 58 57 56 55 54 53 52 51

41 42 43 44 45 46 47 48 49 50

40 39 38 37 36 35 34 33 32 31

21 22 23 24 25 26 27 28 29 30

20 19 18 17 16 15 14 13 12 11

1 #P1 3 4 5 6 7 8 9 10
```

```
Player 1, press Enter to roll the die...You rolled a 2.
Player 1 is now at square 95.

100 99 98 97 96 #P1 94 93 92 91

81 82 83 84 85 86 87 88 89 90

80 79 78 77 76 75 74 73 72 71

61 62 63 64 65 66 67 68 69 70

60 59 58 57 56 55 54 53 52 51

41 42 #P2 44 45 46 47 48 49 50

40 39 38 37 36 35 34 33 32 31

21 22 23 24 25 26 27 28 29 30

20 19 18 17 16 15 14 13 12 11

1 2 3 4 5 6 7 8 9 10

Player 2, press Enter to roll the die...You rolled a 6.
Player 2 is now at square 49.

100 99 98 97 96 #P1 94 93 92 91

81 82 83 84 85 86 87 88 89 90

80 79 78 77 76 75 74 73 72 71

61 62 63 64 65 66 67 68 69 70

60 59 58 57 56 55 54 53 52 51

41 42 43 44 45 46 47 48 #P2 50

40 39 38 37 36 35 34 33 32 31

21 22 23 24 25 26 27 28 29 30

20 19 18 17 16 15 14 13 12 11

1 2 3 4 5 6 7 8 9 10
```

```
Player 1, press Enter to roll the die...You rolled a 5.
Player 1 is now at square 100.

#P1 99 98 97 96 95 94 93 92 91

81 82 83 84 85 86 87 88 89 90

80 79 78 77 76 75 74 73 72 71

61 62 63 64 65 66 67 68 69 70

60 59 58 57 56 55 54 53 52 51

41 42 43 44 45 46 47 48 #P2 50

40 39 38 37 36 35 34 33 32 31

21 22 23 24 25 26 27 28 29 30

20 19 18 17 16 15 14 13 12 11

1 2 3 4 5 6 7 8 9 10
```

Player 1wins!


Thank
you!