PROJECT TITLE

{Snake And Ladder Game}

Student Name: Sumit Sharma **UID: 24BCA10457**
Branch: BCA **Section/Group: 24BCA7A**
Semester: First Semester **Date of Performance: 24 /10/2024**
Subject Name: C PROGRAMMING **Subject Code: 24CAH-101**

1. **Aim/Overview of the practical:**
   To create a game alike snake ladder using c programming language.

2. **Task to be done:**

- Define the game rules and flow, including player turns and win conditions.
- Install and configure a C compiler (e.g., GCC) and a code editor (e.g., VS Code).
- Write functions to roll the dice, move players, and manage game states (snakes and ladders).
- Develop a function to visually represent the game board with appropriate markings for players, snakes, and ladders.
- Implement input prompts for player turns and actions.
- Run multiple test cases to ensure all game mechanics work as intended (e.g., correct movement, win detection).
- Optimize the code for readability and performance, adding comments to explain key sections.

3. Algorithm/Flowchart :

- Initialize global variables to track player positions (player1, player2) and define arrays for snakes and ladders.

- Create a function to display the game board, marking player positions and indicating snakes (downward arrows) and ladders (upward arrows).

- Implement a function to simulate rolling a six-sided die.

- In a loop, prompt the current player for input, roll the dice, and update the player's position based on the rolled number.

- Check for snakes or ladders at the new position and adjust the player's position accordingly.

- After each move, check if the player has reached square 100. If so, declare them as the winner and reset the game.

- Alternate turns between the two players until a winner is determined.

1. Code for experiment/practical:

```c
// 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()
{
    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;

    int ladder[] = {6, 77};  // Example ladder positions
    int mouth[] = {23, 45, 61, 65, 98};  // Example snake positions

    while (val--) {

        if (alt == 0) {
            iterLR--;
            if (iterLR == player1) {
                printf("(P1) ");

            }
            else if (iterLR == player2) {
                printf("(P2) ");

            }
            else {
                // Check if the current board position is a ladder or snake
                int isLadder = 0, isSnake = 0;

                // Check if the board position is in the ladder array
                for (int i = 0; i < sizeof(ladder)/sizeof(ladder[0]); i++) {
                    if (board[iterLR] == ladder[i]) {
                        isLadder = 1;
                        break;
                    }
                }

                // Check if the board position is in the mouth (snake) array
                for (int i = 0; i < sizeof(mouth)/sizeof(mouth[0]); i++) {
                    if (board[iterLR] == mouth[i]) {
                        isSnake = 1;
                        break;
                    }
                }

                // Print the board number and the appropriate arrow
                printf("%d", board[iterLR]);
                if (isLadder) {
                    printf("|| ");  // Ladder position (arrow up)
                } else if (isSnake) {
                    printf("~ ");  // Snake position (arrow down)
                } else {
                    printf("  ");  // Normal position

                }
            }

            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 {
        // Check if the current board position is a ladder or snake
        int isLadder = 0, isSnake = 0;

        // Check if the board position is in the ladder array
        for (int i = 0; i < sizeof(ladder)/sizeof(ladder[0]); i++) {
            if (board[iterRL] == ladder[i]) {
                isLadder = 1;
                break;
            }
        }

                // Check if the board position is in the mouth (snake) array
                for (int i = 0; i < sizeof(mouth)/sizeof(mouth[0]); i++) {
                    if (board[iterRL] == mouth[i]) {
                        isSnake = 1;
                        break;
                    }
                }

                // Print the board number and the appropriate arrow
                printf("%d", board[iterRL]);
                if (isLadder) {
                    printf("|| ");  // Ladder position (arrow up)
                } else if (isSnake) {
                    printf("~ ");  // Snake position (arrow down)
                } else {
                    printf("  ");  // Normal position
                }
            }

            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.
```

```c
    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()
{
  int currentPlayer = 1;
  int won = 0;
  int number;

    // Prompt the user for input


    // Use scanf to read the input

    srand(time(0)); // Initialize random seed


    printf("Snake and Ladder Game\n");

    while (!won) {
        printf("Enter a player Name: ");
        scanf("%d", &currentPlayer);
        getchar(); // Wait for the player to press Enter
        int roll = rollDie();


        if (currentPlayer == 1) {
            printf("You rolled a %d.\n", roll);
            system("cls");
            player1 = movePlayer(player1, roll);
            printf("You have got %d\n",roll);
            printf("Now you are at position %d.\n\n",
                player1);

            printBoard();
            if (player1 == 100) {
                printf("Player 1 wins!\n\n\n");
                won = 1;
                player1 = 0;
                main();
            }
        }
        else if (currentPlayer == 2) {
```

```c
                    printf("You rolled a %d.\n", roll);
                    system("cls");
                    player2 = movePlayer(player2, roll);
                    printf("You have got %d\n",roll);
                    printf("Now you are at position %d.\n\n",
                        player2);

                    printBoard();
                    if (player2 == 100) {
                        printf("Player 2 wins!\n");
                        won = 1;
                        player2 = 0;
                        main();
                    }
                } else{
                    printf("Invalid player id i.e  %d",currentPlayer);
                    printf("\n");
                }

                // Switch to the other player
                currentPlayer = (currentPlayer == 1) ? 2 : 1;
            }

            return 0;
        }

```

2. Result/Output/Writing Summary:

- **Dice Rolling**: Players roll a six-sided die to determine their move.
- **Player Movement**: Each player moves forward based on the dice roll, encountering snakes and ladders that affect their positions:

- **Snakes**: Landing on a snake sends the player backward.
- **Ladders**: Landing on a ladder propels the player forward.

- **Board Representation**: The game board is printed dynamically, showing player positions and indicating snakes and ladders with specific symbols.
- **Winning Condition**: The game continues until a player reaches square 100, at which point they are declared the winner.
- **User Input**: Players enter their identifiers, and the program prompts them for their turns, making it interactive.

Learning outcomes (What I have learnt):

- Gain an understanding of basic C programming concepts, including variables, loops, and conditional statements.
- Develop skills in implementing random number generation for simulating dice rolls.
- Learn how to use arrays to manage and represent game elements, such as snakes and ladders.
- Understand how to create and call functions to organize code and improve readability.
- Acquire experience in using user input to control game flow and player interactions.
- Enhance problem-solving skills by handling game rules and win conditions.
- Improve debugging skills by troubleshooting issues related to player movement and board representation.

Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Demonstration and Performance (Pre Lab Quiz)		5
2.	Worksheet		10
3.	Post Lab Quiz		5