# MINI PROJECT

*Submitted by*

**Sangam Smilika Reddy (RA2111032010025)**
**Vaishnavi Kumari (RA2111032010017)**
**Balaram Reddy (RA2111032010012)**

*Under the Guidance of*

**Dr. A.Prabhu Chakkaravarthy**

**Assistant Professor, Department of Networking and Communitcation**

*In partial satisfaction of the requirements for the degree of*

**BACHELORS OF TECHNOLOGY**
**in**
**COMPUTER SCIENCE ENGINEERING**

**with specialization in Internet Of Things**


**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR – 603203**

**May 2023**

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

# BONAFIDE CERTIFICATE

Certified that the Course 18CSE204J-Design and Analysis of Algorithm Assignment Report titled **"SNAKE AND LADDER WITH OPTIMISED MINIMUM MOVES TO WIN"** is the bonafide work done by **SANGAM SMILIKA REDDY [RA2111032010025], VAISHNAIVI KUMARI [RA2111032010017] and BALARAM REDDY[RA21110320100** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

**SIGNATURE**
Faculty In-Charge
**Dr.A.Prabhu Chakkaravarthy**
Assistant Professor
Department of Networking and Communitcations,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

**HEAD OF THE DEPARTMENT**
**Dr. Annapurani Panaiyappan K**
Professor and Head ,
Department Networking and Communitcations
Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

## Index



## Teamwork Plan & Execution

| Team Member Name | Work Completed | Remarks |
|---|---|---|
| Sangam Smilika Reddy | Implementation of Algorithm and error free code | |
| Vaishnavi Kumari | Implementation of Algorithm and error free code | |
| Balaram Reddy | Implementation of Algorithm and error free code | |

**Team Member 1 Sign** **Team Member 2 Sign** **Team Member 3 Sign**

# Snakes & Ladders

## Introduction

The Snakes & Ladders game is a popular board game that has been enjoyed by players for many generations. It is a game of chance that involves rolling dice and moving pieces around a board, trying to reach the finish line before the other players. In this project, we have implemented the Snakes & Ladders game in C++ using BFS to compute the minimum number of moves required to win. The game challenges the player to win in the minimum number of moves possible.

## Technologies

This project has been implemented in C++ programming language. The BFS algorithm has been used to compute the minimum number of moves required to win the game.

## Features

Randomly generates the positions of the snakes and ladders on the board
Allows the player to roll dice to move their piece around the board
Computes the minimum number of moves required to win the game using BFS
Challenges the player to win in the minimum number of moves possible

## How to Play

1. Compile and run the C++ code
2. The game will randomly generate the positions of the snakes and ladders on the board
3. The player can roll dice to move their piece around the board
4. The game will keep track of the player's moves and the minimum number of moves required to win
5. The game will end when the player reaches the finish line
6. The player is challenged to win in the minimum number of moves possible

## Implementation

- The board is represented using a vector of integers, where each element represents a square on the board.
- The positions of the snakes and ladders on the board are randomly generated using the rand() function.
- The BFS algorithm is used to compute the minimum number of moves required to win the game. The algorithm uses a queue to keep track of the squares that need to be explored and a vector to keep track of the squares that have already been explored.
- The game allows the player to roll dice by generating a random number between 1 and 6 using the rand() function.
- The game keeps track of the player's moves and the minimum number of moves required to win using integer variables.

## Code

```
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

// Board class for representing the Snakes & Ladders game board
class Board {
private:
    vector<int> squares;
public:
    Board() {
        squares.resize(100, 0);
    }
    void add_snake(int start, int end) {
        squares[start-1] = end-1;
    }
    void add_ladder(int start, int end) {
        squares[start-1] = end-1;
    }
    int get_destination(int square) {
        return squares[square];
    }
};
// BFS function to compute the minimum number of moves required to win
int bfs(Board& board) {
    queue<int> q;
    vector<int> visited(100, 0);
    q.push(0);
    visited[0] = 1;
    int moves = 0;
    while (!q.empty()) {
        int size = q.size();
        while (size--) {
            int curr = q.front(); q.pop();
            if (curr == 99) {
                return moves;
            }
            for (int i = 1; i <= 6; i++) {
                int next = curr + i;
                if (next < 100 && !visited[next]) {
                    visited[next] = 1;
                    int dest = board.get_destination(next);
                    if (dest != 0) {
                        visited[dest] = 1;
```

```
                    q.push(dest);
                } else {
                    q.push(next);
                }
            }
        }
    }
    moves++;
  }
  return -1;
}
int main() {
  // Create the board and add snakes and ladders
  Board board;
  board.add_snake(17, 7);
  board.add_ladder(4, 14);
  board.add_snake(19, 11);
  board.add_ladder(21, 42);
  board.add_ladder(24, 38);
  board.add_snake(27, 3);
  board.add_ladder(35, 43);
  board.add_snake(50, 5);
  board.add_ladder(55, 65);
  board.add_ladder(62, 96);
  board.add_snake(71, 20);
  board.add_ladder(87, 93);
  board.add_snake(98, 79);

  // Play the game and compute the minimum number of moves required to win
  int moves = bfs(board);
  cout << "Congratulations, you won in " << moves << " moves!" << endl;

  return 0;
}
```

## Conclusion

In this project, we have implemented the Snakes & Ladders game in C++ using BFS to compute the minimum number of moves required to win. The game challenges the player to win in the minimum number of moves possible. This project can be improved and expanded upon by adding features such as player input, graphical user interface, and multiple players.

## Reference


Breadth First Search or BFS for a Graph - GeeksforGeeks
Minimum steps to reach target by a Knight | Set 2 - GeeksforGeeks
DESIGN AND ANALYSIS OF ALGORITHMS (18CS42) (azdocuments.in)