Modern Education Society's Wadia College of Engineering, Pune

210256: DATA STRUCTURES and ALGORITHM LABORATORY (2019 C0URSE)

NAME OF STUDENT:	**CLASS:**
SEMESTER/YEAR:	**ROLL NO:**
DATE OF PERFORMANCE:	**DATE OF SUBMISSION:**
EXAMINED BY:	**EXPERIMENT NO: Mini Project**

TITLE: Design a mini project to implement Snake and Ladders Game using tree data structure.

AIM/PROBLEM STATEMENT:

You are tasked with implementing a console-based Snake and Ladder game using C++. The game should simulate the classic board game where players roll a dice to advance on a board, encountering snakes and ladders along the way.

OBJECTIVES:

1. To understand the concept of tree data structure.
2. To understand concept & features of object-oriented programming.

OUTCOMES:

1. To analyze the working of various Tree operations.
2. Define class for structures using Object Oriented features.

PRE-REQUISITES:

1. Knowledge of C++ programming
2. Basic knowledge of binary search tree ADT
3. Basic knowledge of binary search tree.

APPARATUS:

1. OS: - Ubuntu 20.04 LTS
2. Language: - C++

PROJECT DESCRIPTION:

The given C++ program creates a text-based version of the traditional board game "Snakes and Ladders."

The program consists of three main parts: the Node struct, representing nodes in a Binary Search Tree (BST); the Player class, handling player attributes such as name and current position; and the SnakesAndLadders class, coordinating the game's logic.

At the heart of the game's functionality is the handling of snakes and ladders through charts, where each chart stores the positions of snakes and ladders as key-value pairs.

These structures are then used to construct a BST, enabling efficient search operations during gameplay. Players engage with the game through a series of cues and feedback messages, directing them through the rules and updating them on their progress.

The game loop continues until a player reaches or surpasses the maximum position value (100), at which point the game declares them the champion.

Random dice rolls, executed using the getDiceValue() function, imitate player movement, while the movePlayer() function manages the details of position adjustments based on dice rolls and encounters with snakes or ladders.

Memory management is handled appropriately, with dynamic memory allocation for player items and correct cleanup of BST nodes upon game completion to avoid memory leaks.

The program encapsulates essential concepts of C++ programming, including object-oriented design, data structures, control flow, and memory management, providing a practical and interactive version of the Snakes and Ladders game in a written form.

PROGRAM CODE:-

```
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <map>
#include <string>
using namespace std;
const int MAX_VAL = 100;
const int DICE_FACE = 6;

// Node structure for Binary Search Tree
struct Node {
    int key;
    int value;
    Node* left;
    Node* right;


    Node(int key, int value){
        key = key;
        value = value;
        left = NULL;
        right=NULL;
    }
};

// Player class representing a player in the game
class Player {
private:
    string name;
    int current_position;

public:
    Player(const string& _name) {
        name = _name;
        current_position = 0;
    }

    string getName() const {
        return name;
    }

    int getCurrentPosition() const {
        return current_position;
    }

    void move(int steps) {
        current_position += steps;
```

```
    }
};

// Game class representing the Snakes and Ladders game
class SnakesAndLadders {
private:
    map<int, int> snakes;
    map<int, int> ladders;
    Node* root;
    Player* players[2];

public:
    SnakesAndLadders(const string& player1Name, const string&
player2Name) {
        players[0] = new Player(player1Name);
        players[1] = new Player(player2Name);
        root = NULL;

        // Initialize snakes and ladders
        initializeSnakes();
        initializeLadders();
        // Build BST
        buildBST();
    }

    ~SnakesAndLadders() {
        delete players[0];
        delete players[1];
        clearBST(root);
    }

    void start() {
        welcomeMsg();
        while (true) {
            for (int i = 0; i < 2; ++i) {
                cout << players[i]->getName() << ": " <<
getTurnText() << " Hit Enter to roll dice: ";
                getchar();
                cout << "\nRolling dice..." << endl;
                int diceValue = getDiceValue();
                cout << players[i]->getName() << " moving...."
<< endl;
                movePlayer(players[i], diceValue);
                if (checkWin(players[i])) {
                    cout << players[i]->getName() << " has won
the game. Congratulations!" << endl;
                    return;
                }
            }
```

```
        }
    }

private:
    void welcomeMsg() {
        cout << "\nWelcome to Snakes and Ladders.\n\nRules:\n";
        cout << "1. Initially both the players are at starting
position i.e. 0.\n";
        cout << "2. Take turns to roll the dice.\n";
        cout << "3. Move forward the number of spaces shown on
the dice.\n";
        cout << "4. If you land at the bottom of a ladder, you
can move up to the top of the ladder.\n";
        cout << "5. If you land on the head of a snake, you must
slide down to the bottom of the snake.\n";
        cout << "6. The first player to get to the FINAL
position is the winner.\n";
        cout << "7. Hit enter to roll the dice.\n\n";
    }

    string getTurnText() {
        string turnText[] = {
            "Your turn.",
            "Go.",
            "Please proceed.",
            "Let's win this.",
            "Are you ready?",
            ""
        };
        return turnText[rand() % 5];
    }

    int getDiceValue() {

        srand(time(NULL));
        return rand() % DICE_FACE + 1;
    }

    void initializeSnakes() {
        snakes = {
            {8, 4}, {18, 1}, {26, 10}, {39, 5}, {51, 6}, {54,
36}, {56, 1},
            {60, 23}, {75, 28}, {83, 45}, {85, 59}, {90, 48},
{92, 25}, {97, 87}, {99, 63}
        };
    }

    void initializeLadders() {
        ladders = {
```

```
            {3, 20}, {6, 14}, {11, 28}, {15, 34}, {17, 74}, {22,
37}, {38, 59},
            {49, 67}, {57, 76}, {61, 78}, {73, 86}, {81, 98},
{88, 91}
        };
    }

    void buildBST() {
        for (const auto& snake : snakes) {
            root = insertNode(root, snake.first, snake.second);
        }

        for (const auto& ladder : ladders) {
            root = insertNode(root, ladder.first,
ladder.second);
        }
    }

    Node* insertNode(Node* root, int key, int value) {
        if (root == NULL) {
            return new Node(key, value);
        }

        if (key < root->key) {
            root->left = insertNode(root->left, key, value);
        } else if (key > root->key) {
            root->right = insertNode(root->right, key, value);
        }

        return root;
    }

    void movePlayer(Player* player, int diceValue) {
        int oldPosition = player->getCurrentPosition();
        int newPosition = oldPosition + diceValue;
        cout << "It's a " << diceValue << endl;
        cout << player->getName() << " moved from " <<
oldPosition << " to " << newPosition << endl;

        int finalPosition = searchBST(root, newPosition);
        if (finalPosition != -1) {
            if (finalPosition < newPosition) {
                gotSnakeBite(newPosition, finalPosition, player-
>getName());
            } else {
                gotLadderJump(newPosition, finalPosition,
player->getName());
            }
            player->move(finalPosition - oldPosition);
```

```
        } else {
            player->move(diceValue);
        }
    }

    int searchBST(Node* root, int key) {
        if (root == NULL || root->key == key) {
            return (root == NULL) ? -1 : root->value;
        }

        if (key < root->key) {
            return searchBST(root->left, key);
        } else {
            return searchBST(root->right, key);
        }
    }

    void gotSnakeBite(int oldPosition, int newPosition, const
string& playerName) {
        string messages[] = {
            "boohoo",
            "bummer",
            "snake bite",
            "oh no",
            "dang"
        };
        cout << messages[rand() % 5] << " ~~~~~~~~>\n";
        cout << playerName << " got a snake bite. Down from " <<
oldPosition << " to " << newPosition << endl;
    }

    void gotLadderJump(int oldPosition, int newPosition, const
string& playerName) {
        string messages[] = {
            "woohoo",
            "woww",
            "nailed it",
            "woah",
            "yaayyy"
        };
        cout << messages[rand() % 5] << " ########\n";
        cout << playerName << " climbed the ladder from " <<
oldPosition << " to " << newPosition << endl;
    }

    bool checkWin(Player* player) {
        return player->getCurrentPosition() >= MAX_VAL;
    }
```

```
    void clearBST(Node* root) {
        if (root == NULL) return;
        clearBST(root->left);
        clearBST(root->right);
        delete root;
    }
};

int main() {
    string player1Name, player2Name;
    cout << "Please enter a valid name for the first player: ";
    getline(cin, player1Name);
    cout << "Please enter a valid name for the second player: ";
    getline(cin, player2Name);

    SnakesAndLadders game(player1Name, player2Name);
    game.start();

    return 0;
}
```

OUTPUT:-

Please enter a valid name for the first player: khushal

Please enter a valid name for the second player: Sarthak

Welcome to Snakes and Ladders.

Rules:

1. Initially both the players are at starting position i.e. 0.
2. Take turns to roll the dice.
3. Move forward the number of spaces shown on the dice.
4. If you land at the bottom of a ladder, you can move up to the top of the ladder.
5. If you land on the head of a snake, you must slide down to the bottom of the snake.
6. The first player to get to the FINAL position is the winner.
7. Hit enter to roll the dice.

khushal: Go. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 2
khushal moved from 0 to 2
sarthak: Your turn. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 5
sarthak moved from 0 to 5
khushal: Are you ready? Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 3
khushal moved from 2 to 5
sarthak: Are you ready? Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 3
sarthak moved from 5 to 8
dang ~~~~~~~~>
sarthak got a snake bite. Down from 8 to 4
khushal: Your turn. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 3
khushal moved from 5 to 8
bummer ~~~~~~~~>
khushal got a snake bite. Down from 8 to 4

sarthak: Go. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 3
sarthak moved from 4 to 7
khushal: Go. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 6
khushal moved from 4 to 10
sarthak: Go. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 6
sarthak moved from 7 to 13
khushal: Go. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 6
khushal moved from 10 to 16
sarthak: Go. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 1
khushal moved from 35 to 36
sarthak: Let's win this. Hit Enter to roll dice:

Rolling dice...
sarthak moving....

It's a 1
sarthak moved from 7 to 8
oh no ~~~~~~~~>
sarthak got a snake bite. Down from 8 to 4
khushal: Your turn. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 1
khushal moved from 36 to 37
sarthak: Let's win this. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 1
sarthak moved from 4 to 5
khushal: Let's win this. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 5
khushal moved from 37 to 42
sarthak: Go. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 5
sarthak moved from 5 to 10
khushal: Go. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 5
khushal moved from 42 to 47

sarthak: Go. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 5
sarthak moved from 10 to 15
woww ########
sarthak climbed the ladder from 15 to 34
khushal: Your turn. Hit Enter to roll dice:

Rolling dice...
khushal moving....
It's a 3
khushal moved from 40 to 43
sarthak: Your turn. Hit Enter to roll dice:

Rolling dice...
sarthak moving....
It's a 6
sarthak moved from 94 to 100
sarthak has won the game. Congratulations!