LUDO GAME REPORT

OPERATING SYSTEM PROJECT

Submitted By:

Umar Zeb (22i-1700)
Ali Virk (22i-1625)
Abubakar Sharif (22i-1575)

Pseudo Code:

```
# Include necessary libraries

# Define constants for colors and board size

# Declare global variables

# Function to simulate rolling a die
function rollDie()
    return random number between 1 and 6

# Function to free up tokens
function free()
    for each player
        for each token
            set alive[player][token] to false

# Function to clear the screen
function clear_screen()
    execute system command to clear the screen

# Function to initialize the board
function initializeBoard()
    # Display the game board with tokens and players
    for each row and column of the board
        print corresponding design character
    print tokens and players in their respective positions on the board

# Function to get the token number at a given position
function getpos(position)
    for each player
        for each token of the player
            if token's position matches given position and the token is alive
                return the token number
    return 0 if no token is found at the given position


Define a class called Player:
    Define a method called moveToken(player, goti, diceRoll):
```

```
if player == 0:
    if counter[player][goti] > 50:
        Set home[player][goti] to true
        if counter[player][goti] > 56:
            Subtract diceRoll from counter[player][goti]
            Clear the screen
            Initialize the game board
        else if counter[player][goti] is equal to 56:
            Print "Player 1's Goti: goti + 1 Has passed"
            Decrement countg[0]
            Set pl1[final[player][goti]] to '0'
        else:
            Set pl1[final[player][goti]] to '0'
            Set final[player][goti] to counter[player][goti] - 51
            Set pl1[final[player][goti]] to '1'
            Clear the screen
            Initialize the game board
    else if token[player][goti] >= 51:
        if token[player][goti] is one of 0, 8, 13, 26, 16, 34, 39, 47:
            Set design[token[player][goti]] to '1'
            Clear the screen
            Initialize the game board
        else:
            if design[token[0][goti]] is '2':
                Set pos to token[player][goti]
                Set p to getpos(pos)
                if p:
                    Set alive[1][p] to false
                Clear the screen
                Initialize the game board
                Print "Player 1 KILLED Player 2"
            else if design[token[0][goti]] is '3':
                Set pos to token[player][goti]
                Set p to getpos(pos)
                if p:
                    Set alive[2][p] to false
                Clear the screen
                Initialize the game board
                Print "Player 1 KILLED Player 3"
            else if design[token[0][goti]] is '4':
```

```
                Set pos to token[player][goti]
                Set p to getpos(pos)
                if p:
                    Set alive[3][p] to false
                Clear the screen
                Initialize the game board
                Print "Player 1 KILLED Player 4"
            else:
                Clear the screen
                Initialize the game board
    else if player == 1:
        Repeat the same logic as above with appropriate changes for player 1
    else if player == 2:
        Repeat the same logic as above with appropriate changes for player 2
    else if player == 3:
        Repeat the same logic as above with appropriate changes for player 3

Define a function called showtokens(player):
    Set a to 0
    For each token in range 1 to 4:
        If alive[player][i] is true:
            Print i
            Increment a by 1
    If a is equal to 0:
        Print "No Tokens Available! :("

Define a function called playerRoutine(arg):
    Set play to arg
    Create a new instance of Player called p1
    Wait for the semaphore hello
    Generate a random dice roll using rollDie() and store it in diceRoll
    Print the dice roll
    Print "Turn of Player: play + 1"
    If diceRoll is equal to 6:
        Print "If you want to spawn a new token then select any token."
        Print "Else you can choose the token number you wanna proceed with:"
        Call showtokens(play)
    Set c to diceRoll
    Set a to 0
    If diceRoll is not equal to 6:
```

```
    For each token in range 0 to 4:
        If alive[play][i] is true:
            Increment a by 1
If a is greater than 0 or diceRoll is equal to 6:
    Repeat:
        Print "Chose Token:"
        Read goti from input
        If goti is less than 0 or goti is greater than 4:
            Print "Invalid token number. Please choose a valid token number (1-4)."
            Continue to the next iteration of the loop
        If alive[play][goti] is false and diceRoll is not equal to 6:
            Print "Chosen token is not alive. Please choose an alive token."
            Continue to the next iteration of the loop
        If play is equal to 0 and alive[0][goti] is false and diceRoll is equal to 6:
            Increment num6[0]
            Set alive[0][goti] to true
            Set diceRoll to 0
            Call p1.moveToken(play, goti - 1, diceRoll)
        Else if play is equal to 1 and alive[1][goti] is false and diceRoll is equal to 6:
            Increment num6[1]
            Set alive[1][goti] to true
            Set diceRoll to 0
            Call p1.moveToken(play, goti - 1, diceRoll + 13)
        Else if play is equal to 2 and alive[2][goti] is false and diceRoll is equal to 6:
            Increment num6[2]
            Set alive[2][goti] to true
            Set diceRoll to 0
            Call p1.moveToken(play, goti - 1, diceRoll + 26)
        Else if play is equal to 3 and alive[3][goti] is false and diceRoll is equal to 6:
            Increment num6[3]
            Set alive[3][goti] to true
            Set diceRoll to 0
            Call p1.moveToken(play, goti - 1, diceRoll + 39)
        Else:
            If alive[play][goti] is true:
                Add diceRoll to counter[play][goti - 1]
                Call p1.moveToken(play, goti - 1, diceRoll)
Else:
    Print "No active tokens to move."
Release the semaphore hello
```

Define the Function fro Checking block condition:
  Iterates the positions
 Finds the position with with two tokens of a player.

Define the main function:
    Seed the random number generator
    Initialize game state variables
    Initialize semaphores
    Set play to the result of calling player()
    Create a master thread using the MasterThread function
    Forever:
        Shuffle the order of players
        For each player in range 0 to 3:
            Create a new thread for the player using the playerRoutine function with the player's index
        For each player in range 0 to 3:
            Join the player thread