

***Project***

**Operating Systems**

**Group Details:**

* Hamza Iftikhar i192003
* Hamza Nasir i190700
* Section D

**Pseudo Codes:**

**Phase 1:**

Grid[]

Dice

function Main():

Initialize window

Initialize grid positions

Create board using DrawRectangle function of raylib

while (Not Exitting):

DrawBoard()

DrawDice()

**Phase 2:**

Grid[]

Dice

function Main(): // Called First

Take total tokens as input

Take player names as input

Initialize window

Create Master thread and then detach

Initialize absolute positions of grid

While (Not Exitting):

DrawBoard();

DrawPieces();

DrawDice();

function MasterThread(): // Called from Main()

Create the player threads.

Detach the player threads

While (True):

Loop i, from 0 to 3:

if (players remaining tokens == 0):

update positions

cancel player thread

bring next players turn

if (All players done):

End game

function PlayerThread(the ‘i’ variable): // Called from Master Thread

While (True):

sem\_wait(DiceSemaphore)

RollDice()

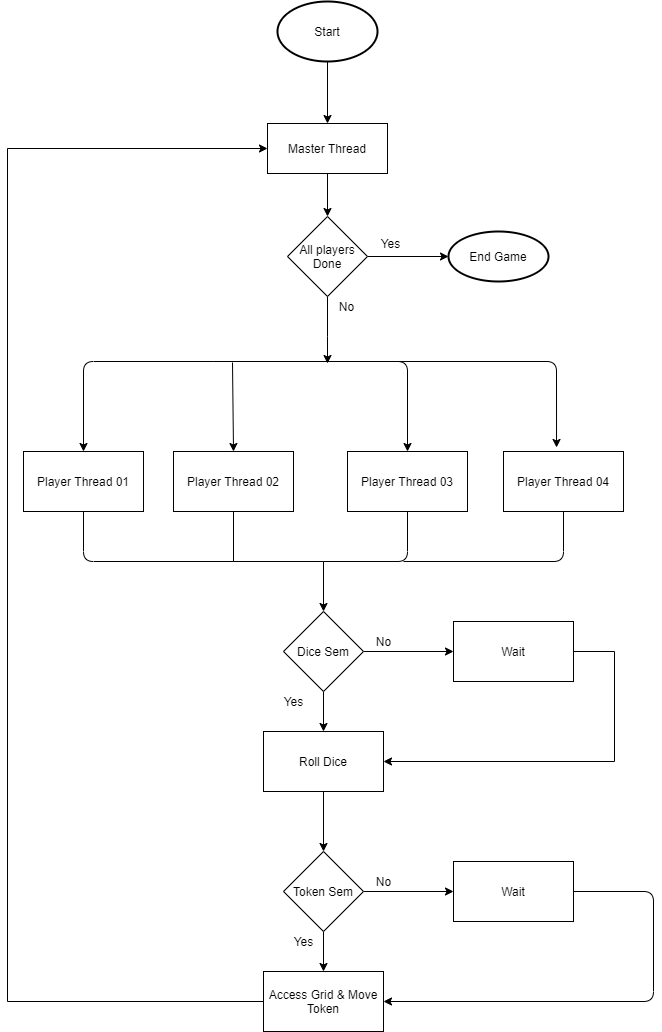
sem\_post(DiceSemaphore)

sem\_wait(TokenSemaphore)

Move Token() // Function to access grid and move tokens

sem\_post(TokenSemaphore)

**Illustrations of OS Concepts used in Pseudo Codes:**



**Implemented Codes:**

All implemented code files are attached along with report.

**Group Participation:**

**1) Hamza Nasir –** Primary game implementation, along with some work on threading.

**2) Hamza Iftikhar –** Primary work on threading, along with some bug and logic fixing on game implementation.

**System Specifications:**

**Hamza Nasir, i190700 –** Intel Core i5-5200U, 2 Cores , 4 threads, 128 HDD, 256 SSD, 64 bit OS, 12 GB RAM

**Hamza Iftikhar, i192003 –** AMD Ryzen 7 3700U, 4 Cores, 8 threads, 256 SSD, 64 bit OS, 12 GB RAM

**How we could implement these concepts in a different scenario**