## horizontal line



Board Game LUDO

05.01.2024

Implementation of LUDO board game via Data Structures and Algorithms.

**────────────────────────────────────**

By Team:

Rehan-Ur-Rehman Sharif | CT - 084

Maha Jameel | CT - 055

Muhammad Taqi Haider | CT - 092

# Project Overview

This Project is an implementation of the **LUDO** board game that showcases the use cases of several data structures and algorithms in order to make the project functional and facilitate its mechanics.

# Key Concepts from DSAA

This Project utilizes the concepts from **SIngly CIrcular Linked Lists** in order to simulate a proper LUDO board, albeit with some modifications to accommodate the forks in the road near “home” lines.

**Traversals** of said linked lists are also utilized for piece movements, and optimized algorithms are used to keep track of pieces and capture states when 2 pieces collide.

# Goals

Create a functional, playable, version of the game in our project.

Implement features through the use of data structures that match our requirements depending on the situation.

Creation of a demonstrable program that the users can easily understand and interact with.

Modular approach using Classes, Libraries and other approaches utilizing OOP and DSAA concepts as well.

# 

# Specifications

Several DSAA specific and Board game Specific mechanics were implemented within our game in order to make it stand out a bit in functionality Some of the notable points of our project are as follows:

## Modular approach

Creation of multiple libraries that serve a specified purpose such that they can be used as in other projects of the same type and reduce the total amount of diverse code at the same time and it also helps in separating code by its specific functionalities.

## Data Structures

Use of custom functionality linked list and structure to simulate a Ludo board properly as well as be able to handle game logic correctly throughout the duration of the game.

## Algorithmic Approach

Creation of a somewhat optimized algorithm for handling specific conditions and scenarios that occur in a ludo board. Optimization of algorithms such that unnecessary calculations are kept at minimum as allowed by our skill.

**Pros**

* Board Shown in color with each move
* Accurate track and updation of locations as pieces move
* Proper Capture logic for players to take each other’s pieces mid play
* Invalid Move Logics implemented to prevent Standstills and Infinite Loops.
* Invalid selection of Moves and other validations.s

**Cons**

* Movement of pieces not shown by color on the board
* Having to keep track of pieces through the Relative positions from their starts.
* Unclear to know position of pieces until it's your turn, due to no colors

# Milestones

## Game Logic

Implemented game initialization logic and movement logic for pieces on the board.Implemented win condition and basic checks for each move using random number generation.

## Board Display (Basic)

Implemented a dysfunctional board to be displayed in order to get a grasp of how the board will look near completion.

## Capture Logic

Implementation of logic to handle piece interaction in the same position that results in the capture move and the relocation of pieces into a jail.

## Piece Independence

Created method to allow players to independently choose which piece they want to move.

## Coloured Board

Use of Windows.h library to recolour console to give a coloured look to the ludo board.

## Computer Controlled Player

Use of random number generation in order to create computer controlled players for automatic playing.

## Final Completion

Integration of individual modules in order to complete the finalized game. That is easily playable by people.

# Workloads

## Rehan-ur-Rehman Sharif

Implemented game initialization logic and movement logic for pieces on the board.Implemented win condition and basic checks for each move using random number generation. Handle LudoBoard.h and LudoGame.h libraries implementation using Ludo Players and Ludo piece library developed earlier by the team.

Coded the Display of the Board and Title.

Went through Integration hell in rounding the entirety of the project.

Overlooked creation of the project report and project presentation and final checking of code and documentation.

## Maha Jameel

Implemented win conditions and piece checking algorithm (optimized) for each turn on the board.Implementation of computer control players. Define the final layout of Ludo board.Handle implementation of LudoPiece.h and Ludo node.h library and piece movement logic and nodes for creation of board later on.

Most (if not all) aesthetic decisions were from here.

Creation and handling of project reports and proofreading of documents.

## Muhammad Taqi Haider

Implementation of logic to handle piece interaction in the same position that results in the capture move and the relocation of pieces into a jail.Collaboration in LudoPlayer.h for turns and other logics.

Testing for failure cases for moves during the game.

Helped in tracking several of the many bugs encountered during testing

Creation of project presentation and the idea for layered libraries.

# Challenges

We faced numerous challenges during our coding of ludo as our semester project due to the complexity of the game that lies beneath its simple outer shell.

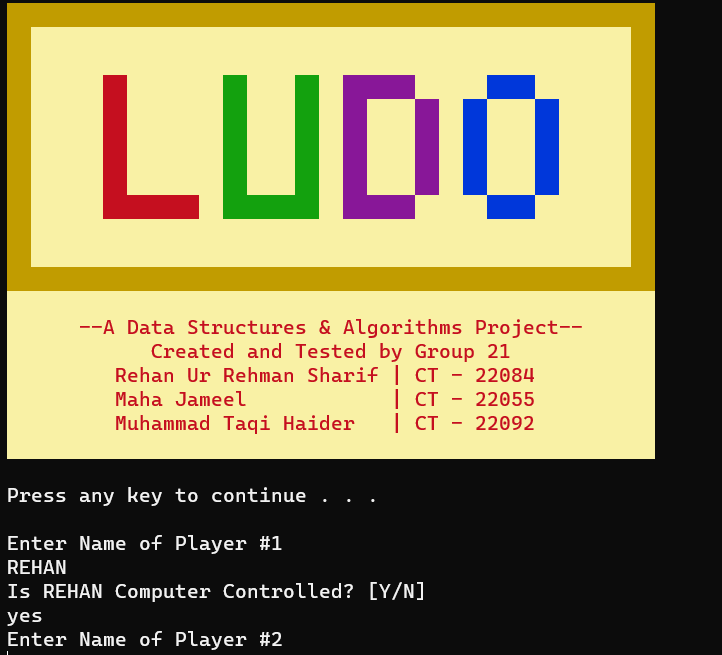
When looking deeper into the logistics of the game and how they could possibly be implemented in a computer program, many problems arose.

FIrstly, The generation of the ludo board through the use of Linked List was first thought, but then having separate locations each having very specific functionalities was our first major block. It was resolved via creating derived classes from our starting Node class to match our needs.

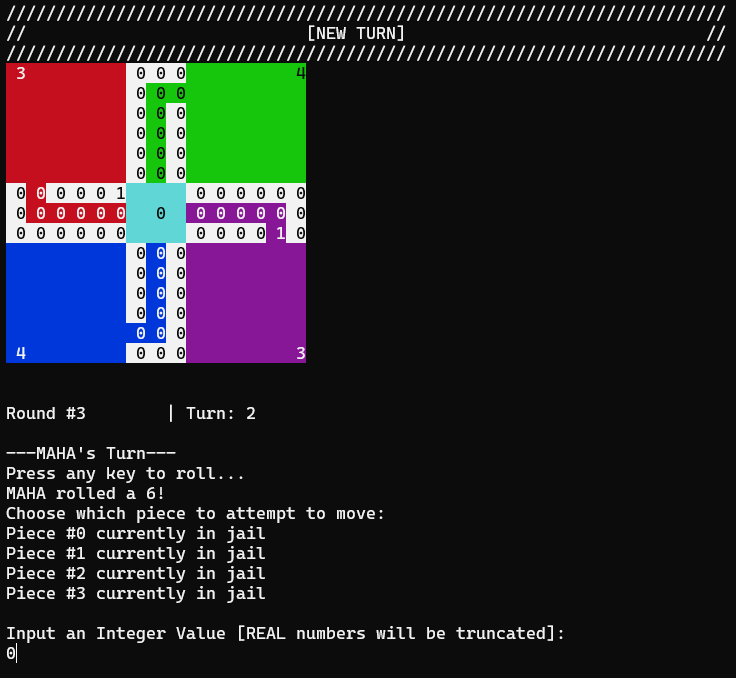
Afterwards, Implementation of piece logic was getting difficult due to not being able to keep track of memory locations of roughly 84 locations at once, due to incorrect algorithms.

Similarly, The conversion and coordination of having the board be displayed in a correct manner on the console by reading values from the linked list was too convoluted to be able to be implemented properly and was resolved through adding an array to store said data for later use.

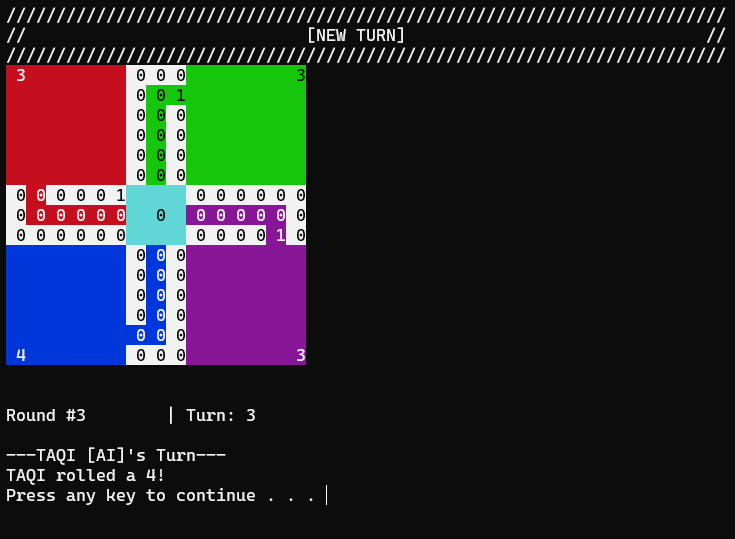
# Outputs

**Main Menu:**

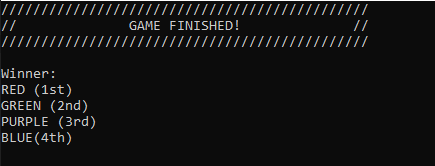
**Example Turn:**

****

**After Turn:**

****

**Winner Screen:**

****