FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

Data Structure Project

Doç. Dr. Berna KİRAZ

Treasure Hunt Adventure Game

Students:

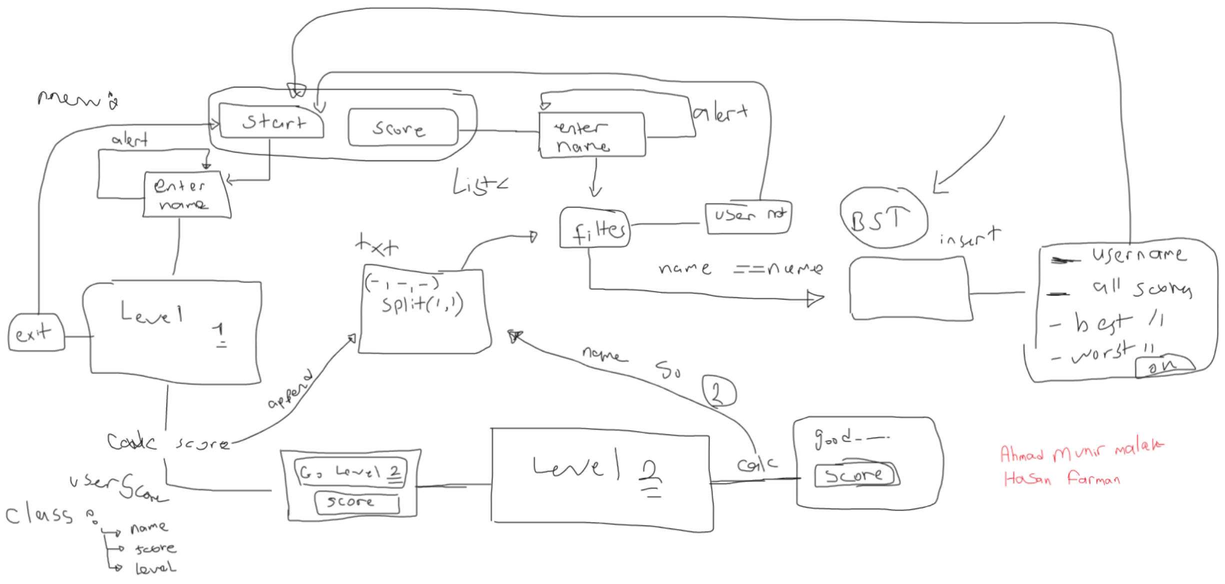
Ahmad Munir Malak 2321021363
Hasan Farman 2321021361

Tutorial In YouTube
https://youtu.be/DAEUF6U1r08

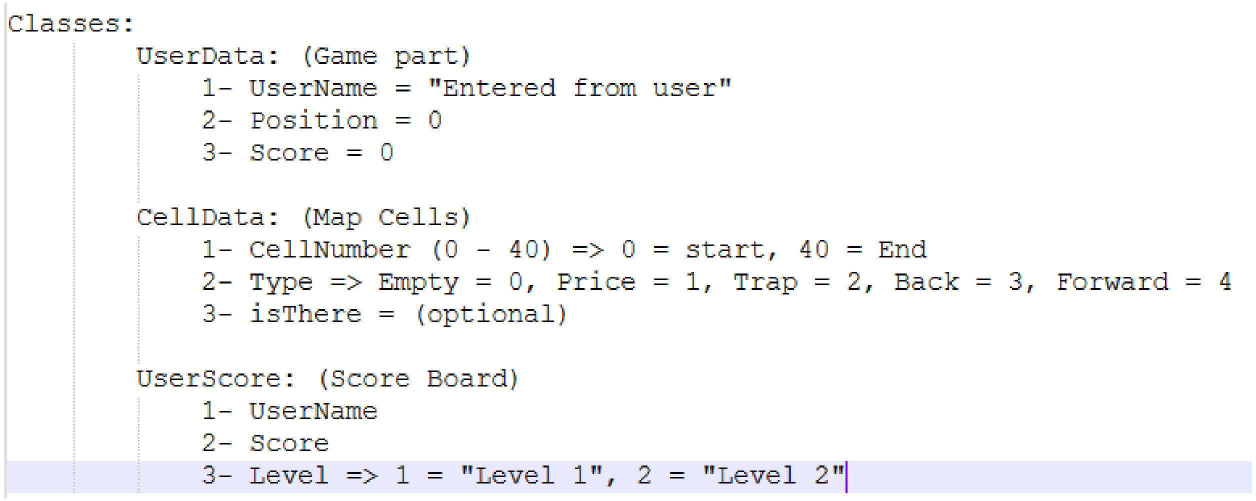
Planning

At the beginning of this project, we held a team meeting to distribute tasks, design the project's architecture, and discuss the core algorithms we would implement. We documented this discussion with a blueprint to guide us toward successful completion.

Blueprint:



Classes:



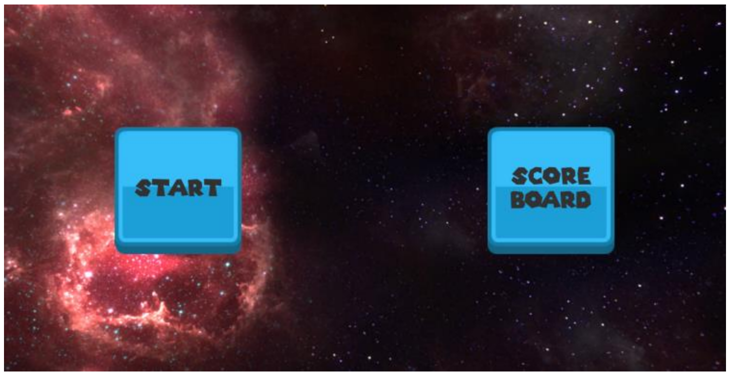
```
Classes:
        UserData: (Game part)
            1- UserName = "Entered from user"
            2- Position = 0
            3- Score = 0

        CellData: (Map Cells)
            1- CellNumber (0 - 40) => 0 = start, 40 = End
            2- Type => Empty = 0, Price = 1, Trap = 2, Back = 3, Forward = 4
            3- isThere = (optional)

        UserScore: (Score Board)
            1- UserName
            2- Score
            3- Level => 1 = "Level 1", 2 = "Level 2"
```

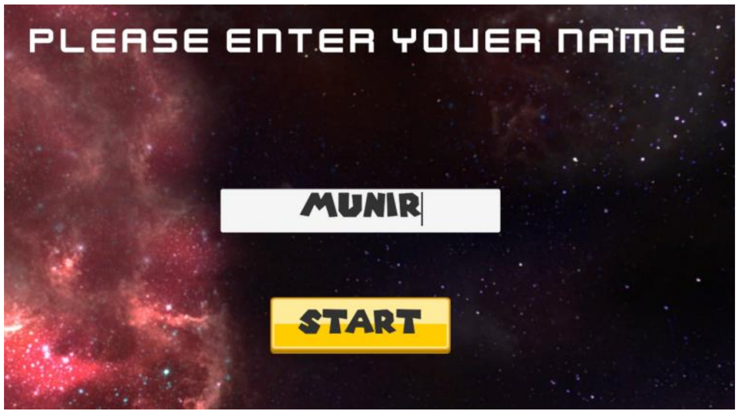
Game Interface

The game starts with a user interface displaying two main buttons:

1. **Start Game**
2. **Score Board**

If the player clicks **Start Game**, they are prompted to enter their name. If they attempt to proceed without entering a name, the system displays a message forcing them to provide one. Once the name is provided, the game begins at **Level1**.





Level 1: The Map

The first map contains **40 cells** and is designed using a **Linked List** structure. To organize and simplify our code, we divided the logic into **methods**. The process flow is:

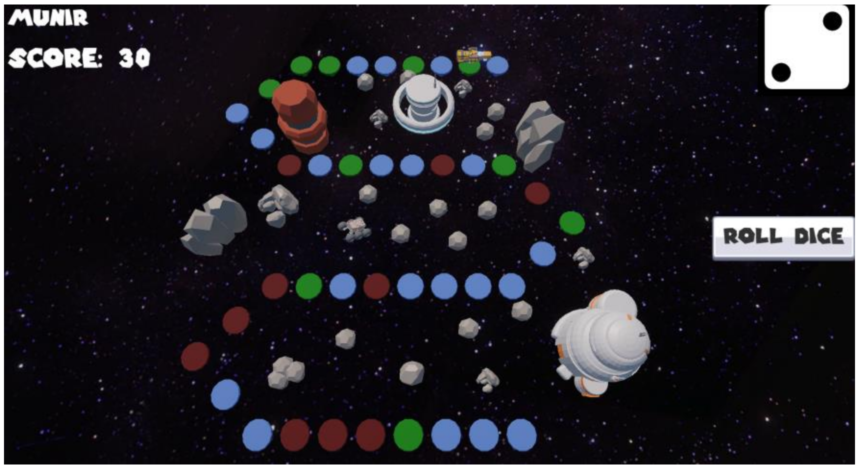
- **First**, the map is created.

- **Then**, treasures and traps are distributed randomly across the cells.
  At each step, we check if a trap or treasure can be placed in a given cell.

Map Details

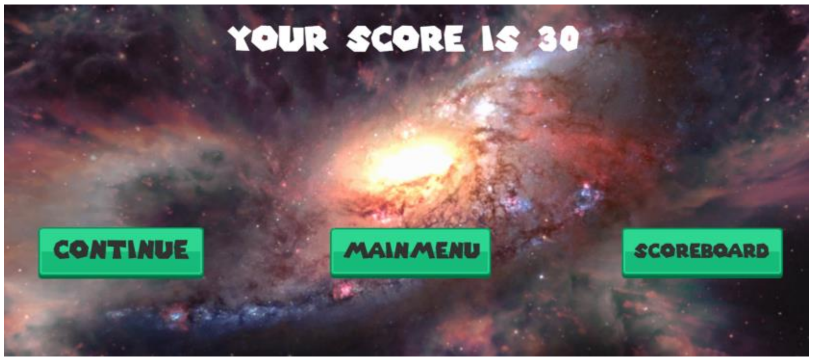
Each cell is represented as a **Node** in the linked list and holds a CellData class. The CellData contains:

- cell index
- cellType:
  - 0 = empty (can hold other types) cellColor,
  - 1 = treasure cellColor,
  - 2 = trap cellColor,
  - 3 = forward cellColor,
  - 4 = backward **cellColor**
- Other properties related to Unity classes



Once the map is prepared, the player's result is saved into a **TXT file**. The player can then roll the dice (randomly generated) and progress through the map. After completing all 40 cells, a screen with **three options** appears:

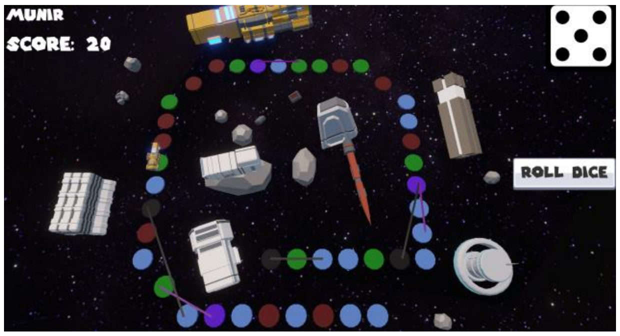
- Proceed to Level 2
- Return to Main Menu
- Go to Results List

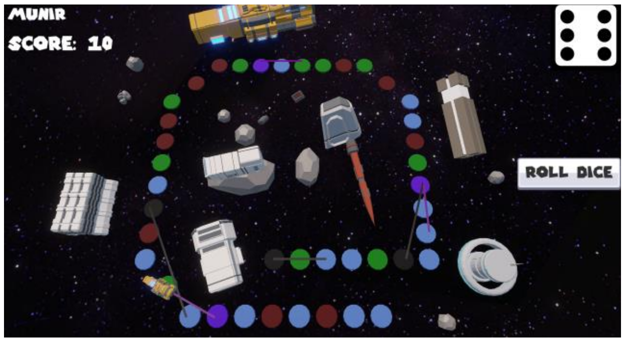


Level 2: The Advanced Map

In Level 2, a new map with **40 cells** is created, but this time using a **DoubleLinkedList**.
Why? Because some cells contain go **forward** or go **backward** instructions, and the double linked list allows smooth navigation without traversing all intermediate cells to reach the target.

After treasures, traps, and movement cells are distributed, the player can enjoy an upgraded and engaging gameplay experience.





Shared Design Through Interfaces

Both the Level 1 and Level 2 maps inherit from a common **interface** to unify their properties while allowing method variations.
For example, both classes share a FindIndex method used during:

- Cell type assignment (initially, all cells are set to empty).
- Handling forward and backward movement.

However, the search strategy differs:

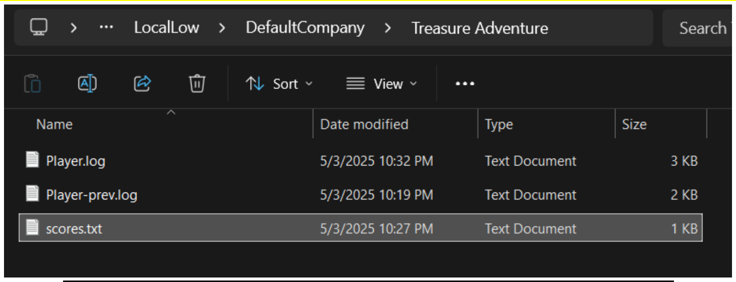
- In the first map (single linked list), the method uses a forward loop (for) from the head to the target cell.
- In the second map (double linked list), the method checks if the target cell is past the halfway point — if so, it loops backward from the tail; otherwise, it loops forward from the head.

After completing Level 2, the data is again saved to the **TXT file**, and the player sees a screen offering two choices:

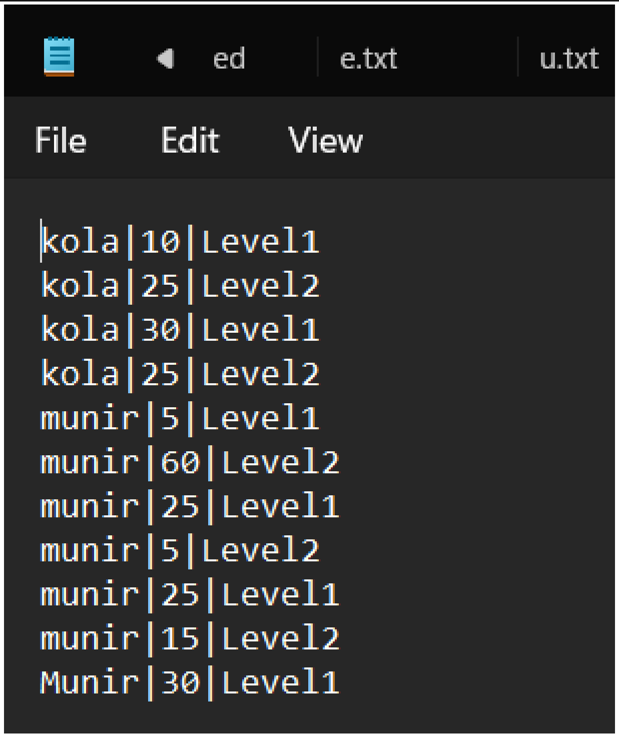
- Start (new Game)
- Score board

Not!

To see the Score.txt file you must follow the path:
C:\Users\{User}\AppData\LocalLow\DefaultCompany\Treasure Adventure



Name	Date modified	Type	Size
Player.log	5/3/2025 10:32 PM	Text Document	3 KB
Player-prev.log	5/3/2025 10:19 PM	Text Document	2 KB
scores.txt	5/3/2025 10:27 PM	Text Document	1 KB



```
kola|10|Level1
kola|25|Level2
kola|30|Level1
kola|25|Level2
munir|5|Level1
munir|60|Level2
munir|25|Level1
munir|5|Level2
munir|25|Level1
munir|15|Level2
Munir|30|Level1
```

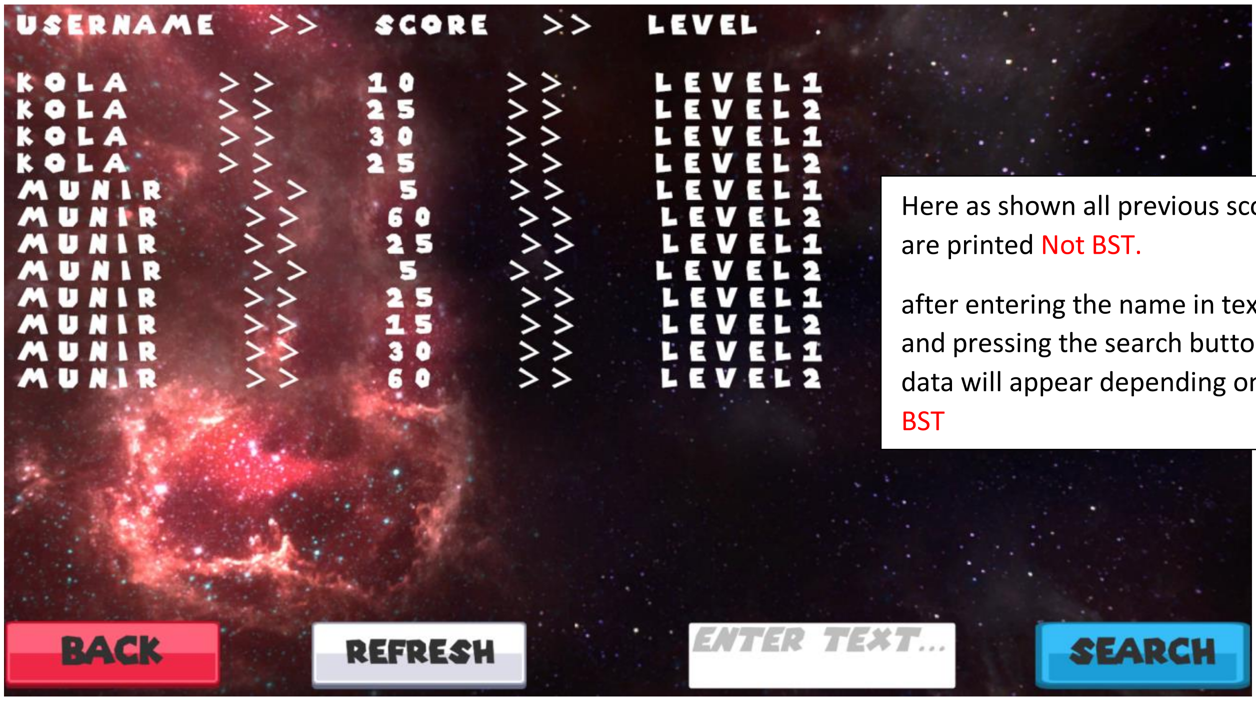
Results List

By default, the results screen loads all past results from the TXT file.
It includes:

- A search field
- A search button
- Back button
- Refresh button

To look up a specific result, the player must enter an **existing name**. The system then:

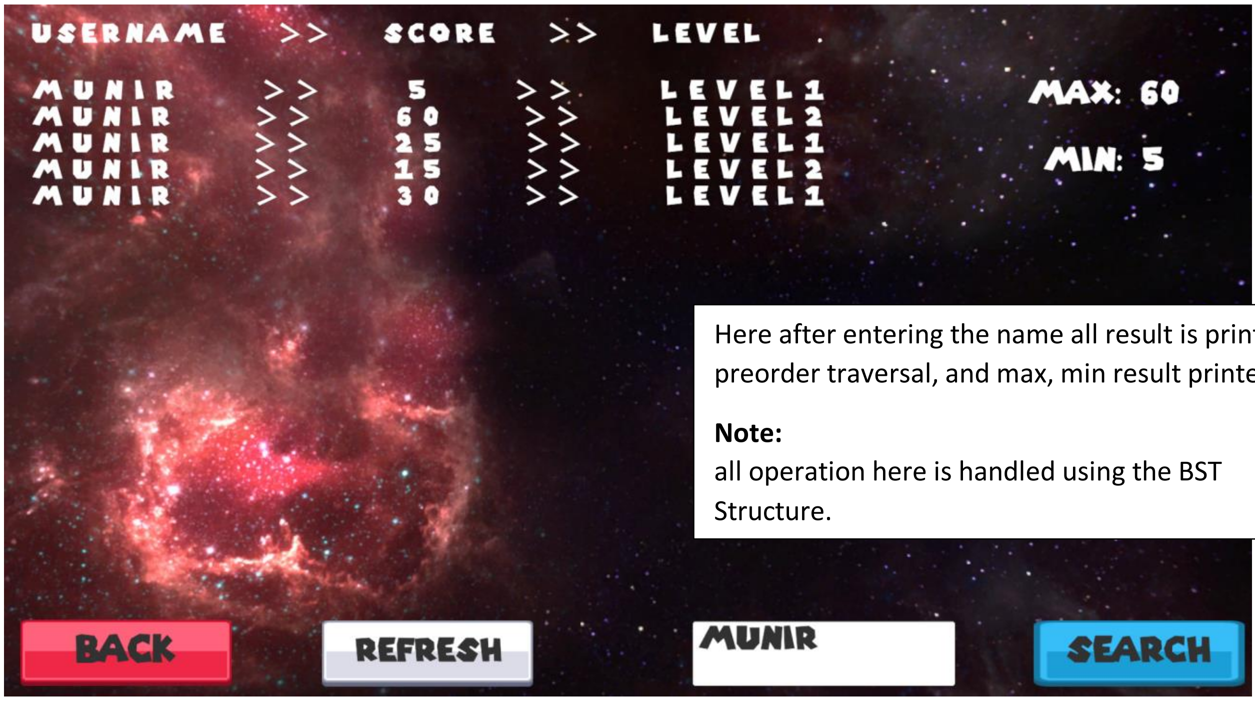
1. Loads all results from the TXT file.
2. Compare each stored name with the entered name.
3. If a match is found that result is inserted into a **Binary Search Tree (BST)**.
4. The system displays the results using **preorder traversal**, showing all records, the **lowest** and **highest** score.



USERNAME	>>	SCORE	>>	LEVEL
KOLA	>>	10	>>	LEVEL1
KOLA	>>	25	>>	LEVEL2
KOLA	>>	30	>>	LEVEL1
KOLA	>>	25	>>	LEVEL2
MUNIR	>>	5	>>	LEVEL1
MUNIR	>>	60	>>	LEVEL2
MUNIR	>>	25	>>	LEVEL1
MUNIR	>>	5	>>	LEVEL2
MUNIR	>>	25	>>	LEVEL1
MUNIR	>>	15	>>	LEVEL2
MUNIR	>>	30	>>	LEVEL1
MUNIR	>>	60	>>	LEVEL2

Here as shown all previous scores are printed Not BST.

after entering the name in text box and pressing the search button, data will appear depending on the BST



USERNAME	>>	SCORE	>>	LEVEL
MUNIR	>>	5	>>	LEVEL1
MUNIR	>>	60	>>	LEVEL2
MUNIR	>>	25	>>	LEVEL1
MUNIR	>>	15	>>	LEVEL2
MUNIR	>>	30	>>	LEVEL1

MAX: 60

MIN: 5

MUNIR

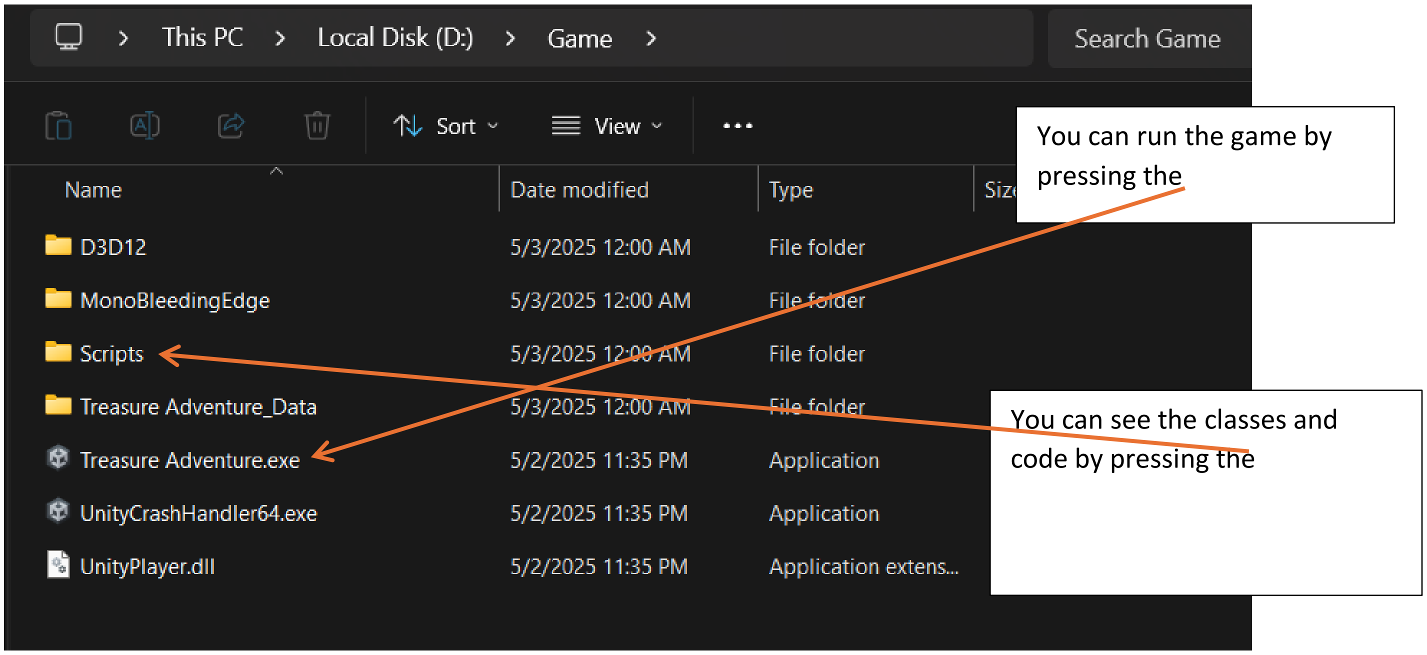
Here after entering the name all result is printed in preorder traversal, and max, min result printed.

Note:
all operation here is handled using the BST Structure.

Project Summary

This project is a game application designed for the Data Structure course, showcasing the practical use of linked lists, double linked lists, and binary search trees. The game features two levels, each built with different data structures to manage a 40-cell map filled with traps, treasures, and movement instructions. Players interact through a simple UI, rolling dice to navigate the map, while their results are saved and later searchable through a results system using a BST for efficient lookups. The project emphasizes structured coding, algorithmic planning, and efficient data handling to deliver an engaging and technically rich game experience.

Some Instruction to run the game and see the code folder:



This PC > Local Disk (D:) > Game >

Name	Date modified	Type	Siz
D3D12	5/3/2025 12:00 AM	File folder	
MonoBleedingEdge	5/3/2025 12:00 AM	File folder	
Scripts	5/3/2025 12:00 AM	File folder	
Treasure Adventure_Data	5/3/2025 12:00 AM	File folder	
Treasure Adventure.exe	5/2/2025 11:35 PM	Application	
UnityCrashHandler64.exe	5/2/2025 11:35 PM	Application	
UnityPlayer.dll	5/2/2025 11:35 PM	Application extens...	

You can run the game by pressing the

You can see the classes and code by pressing the

That was the end of our journey. We hope this report has helped explain our work and the learning we gained through this project. Thank you for your time and attention.

Date: 03/05/2025