MATRIX: CHAR   
 char[,] matrix = new char[sizeOfMatrix, sizeOfMatrix];

int playerRow = 0;

int playerCol = 0;

for (int row = 0; row < sizeOfMatrix; row++)

{

char[] rowValues = Console.ReadLine().ToCharArray();

for (int col = 0; col < sizeOfMatrix; col++)

{

matrix[row, col] = rowValues[col];

if (matrix[row, col] == 'P')

{

playerCol = col;

playerRow = row;

}

}

}

int playerNewRow = playerRow;

int playerNewCol = playerCol;

switch (direction)

{

case "up":

playerNewRow--;

break;

case "down":

playerNewRow++;

break;

case "left":

playerNewCol--;

break;

case "right":

playerNewCol++;

break;

}

matrix[playerNewRow, playerNewCol] = 'P';

matrix[playerRow, playerCol] = '-';

playerRow = playerNewRow;

playerCol = playerNewCol;

playerNewRow >= 0 && playerNewRow < matrix.GetLength(0) && playerNewCol >= 0 && playerNewCol < matrix.GetLength(0)

private static void PrintMatrix(char[,] matrix)

{

for (int row = 0; row < matrix.GetLength(0); row++)

{

for (int col = 0; col < matrix.GetLength(1); col++)

{

Console.Write(matrix[row, col]);

}

Console.WriteLine();

}

}

MATRIX STRING

for (int row = 0; row < sizeOfMatrix; row++)

{

string[] rowValues = Console.ReadLine().Split(" ").ToArray();

for (int col = 0; col < sizeOfMatrix; col++)

{

matrix[row, col] = rowValues[col];

if (matrix[row, col] == "S")

{

playerCol = col;

playerRow = row;

}

}

}

private static void PrintMatrix(string[,] matrix)

{

for (int row = 0; row < matrix.GetLength(0); row++)

{

for (int col = 0; col < matrix.GetLength(1); col++)

{

Console.Write(matrix[row, col]+" ");

}

Console.WriteLine();

}

}

int playerNewRow = playerRow;

int playerNewCol = playerCol;

switch (direction)

{

case "up":

playerNewRow--;

break;

case "down":

playerNewRow++;

break;

case "left":

playerNewCol--;

break;

case "right":

playerNewCol++;

break;

}

matrix[playerNewRow, playerNewCol] = "P";

matrix[playerRow, playerCol] = "-";

playerRow = playerNewRow;

playerCol = playerNewCol;

playerNewRow >= 0 && playerNewRow < matrix.GetLength(0) && playerNewCol >= 0 && playerNewCol < matrix.GetLength(0)

STACK/QUEUE:

int[] boxes = Console.ReadLine().Split(" ", StringSplitOptions.RemoveEmptyEntries).Select(int.Parse).ToArray();

int[] magics = Console.ReadLine().Split(" ", StringSplitOptions.RemoveEmptyEntries).Select(int.Parse).ToArray();

Stack<int> materials = new Stack<int>(boxes); //Last

Queue<int> magic = new Queue<int>(magics); // First

Dictionary<string, int> craftedPresetsTAGS = new Dictionary<string, int>();

materials.Count != 0 && magic.Count != 0

int currentMaterial = materials.Peek();

int currentMagic = magic.Peek();

int multiplication = currentMagic \* currentMaterial;

if (multiplication == 150)

{

craftedPResents.Add("Doll");

if (!craftedPresetsTAGS.ContainsKey("Doll"))

{

craftedPresetsTAGS.Add("Doll", 0);

}

craftedPresetsTAGS["Doll"]++;

materials.Pop();

magic.Dequeue();

}

else if (multiplication == 250)

{

craftedPResents.Add("Wooden train");

if (!craftedPresetsTAGS.ContainsKey("Wooden train"))

{

craftedPresetsTAGS.Add("Wooden train", 0);

}

craftedPresetsTAGS["Wooden train"]++;

materials.Pop();

magic.Dequeue();

}

else if (multiplication == 300)

{

craftedPResents.Add("Teddy bear");

if (!craftedPresetsTAGS.ContainsKey("Teddy bear"))

{

craftedPresetsTAGS.Add("Teddy bear", 0);

}

craftedPresetsTAGS["Teddy bear"]++;

materials.Pop();

magic.Dequeue();

}

else if (multiplication == 400)

{

craftedPResents.Add("Bicycle");

if (!craftedPresetsTAGS.ContainsKey("Bicycle"))

{

craftedPresetsTAGS.Add("Bicycle", 0);

}

craftedPresetsTAGS["Bicycle"]++;

materials.Pop();

magic.Dequeue();

}

magic.Dequeue(); materials.Push(materials.Pop() + 15);

CLASSES

public void Add(Astronaut astronaut)

{

if (data.Count < Capacity)

{

data.Add(astronaut);

}

}

public bool Remove(string name)

{

foreach (var item in data)

{

if (item.Name == name)

{

This.data.Remove(item);

return true;

}

}

return false;

}

public Astronaut GetOldestAstronaut()

{

//return the oldest

Astronaut astronaut = this.data.OrderByDescending(x => x.Age).First();

return astronaut;

}

public Astronaut GetAstronaut(string name)

{

Astronaut astronaut = this.data.Where(x => x.Name == name).FirstOrDefault();

return astronaut;

}