**ALGORITHM AND PROGRAMMING PROJECT 2**

**LOCK GAME**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Furkan İyem……………………….05220000316**

**Altuğ Emre Tosun ………………05220000295**

**CONTENTS**

**- *Programming Catalog***……………………………3

- Data Structure……………………………………………………3

- Functions……………………………………...…………………..4

- ***Test Catalog***

- Test 1……………………………………...…………………………6

- Test 2……………………………………...……………………….11

**Programming Catalog**

**Data Structure**:

Lists:

coordinates = [[' ' for i in range(side\_length)] for j in range(side\_length)]

-This two dimensional list stores the coordinate data.

horizontal=['A','B','C','D','E','F','G','H']

vertical=[1,2,3,4,5,6,7,8]

-These two keep the row’s and column’s names

Dictionaries:

places={'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6,'H':7}

-This stores the index numbers of the rows

places\_inverted = {"0":"A", "1":"B", "2":"C", "3":"D", "4":"E", "5":"F", "6":"G", "7": "H"}

-It only serves to print which stone has been removed

**Functions:**

def main():

-This is the function where the program stars. It gets represents of the players. Then gets the side length of the board. Then creates the coordinates list I have mentioned and calls the new\_board function.

- It also stores the “play\_counter”, “p1\_counter”, “p2\_counter”

**play\_counter:** A counter that is incremented with each turn. It helps determine whose turn it is.

**p1\_counter:** keeps player 1's remaining stones.

**p2\_counter:** keeps player 2's remaining stones.

def new\_board(coordinates,side\_length,play\_counter, p1\_represent, p2\_represent, p1\_counter, p2\_counter):

-Basically it creates and prints the board every single turn then calls the “play” function

- Also checks if the game is over at the end of the function.

def assign\_location(places):

-It gets the current stone’s and target place’s coordinates and converts it to indexes.

def movement\_rules(coordinates, places,play\_counter, p1\_represent, p2\_represent, move\_x, move\_y, stone\_x, stone\_y):

-It checks all the movement rules. If there is any wrong move calls the “assign\_location” function to get coordinates again.

def play(coordinates,side\_length,places,play\_counter, p1\_represent, p2\_represent, p1\_counter, p2\_counter):

-This function prints the names of the players one by one and calls the "assign\_location" function to get the coordinates. Then it calls the "movement\_rules" function and checks for any errors. If it manages to get through, it checks if theres locked stones. If any, it removes the stone.

Finally, it increases the "play\_counter" by one and calls the "new\_board" function to print the current board to the screen.

**Test Catalog**

**Test 1:**

|  |  |  |
| --- | --- | --- |
| input | Expected output | Program output |
| a |  |  |
| b |  |  |
| 5 |  |  |
| 5a 3a |  |  |
| 1b 3b |  |  |
| 5e 3e |  |  |
| 1d 3d |  |  |
| 5c 3c  [(i)](#a" \o "test for removing more than one stone at the same time) |  |  |
| 1e 4e  [(i)](#b" \o "test for jump over a stone) |  |  |
| 1e 1d |  |  |
| 3a 2a |  |  |
| 1d 4d |  |  |
| 5b 1b  [(i)](#c" \o "stone locked in the corner scenario) |  |  |
| 1c 1d |  |  |
| 3e 1e |  |  |
| 4d 5d  [(i)](#d" \o "check for stone in the target cell) |  |  |
| 4d 4c |  |  |
| 5d 5c |  |  |
| y |  |  |

Success status for test1: %100

**Test 2:**

|  |  |  |
| --- | --- | --- |
| input | Expected output | Program output |
| [(i)](#h" \o "invalid character check) |  |  |
| x |  |  |
| x[(i)](#j) |  |  |
| y |  |  |
| 4 |  |  |
| 4c 2c |  |  |
| 1c 2b  [(i)](#e" \o "straight move check) |  |  |
| 1d 3d |  |  |
| 4a 2a |  |  |
| 1b 2b  [(i)](#f" \o "suicide check) |  |  |
| 2c 2d |  |  |
| 1c 4c |  |  |
| 2d 2c  [!!](#g" \o "the previous \"y\" shouldn't been removed but it removed) |  |  |
| n |  |  |

Success status for test2: %93

Download link for inputs:

<https://www.mediafire.com/file/7r9f2vv2fwwwdic/inputs.txt/file>