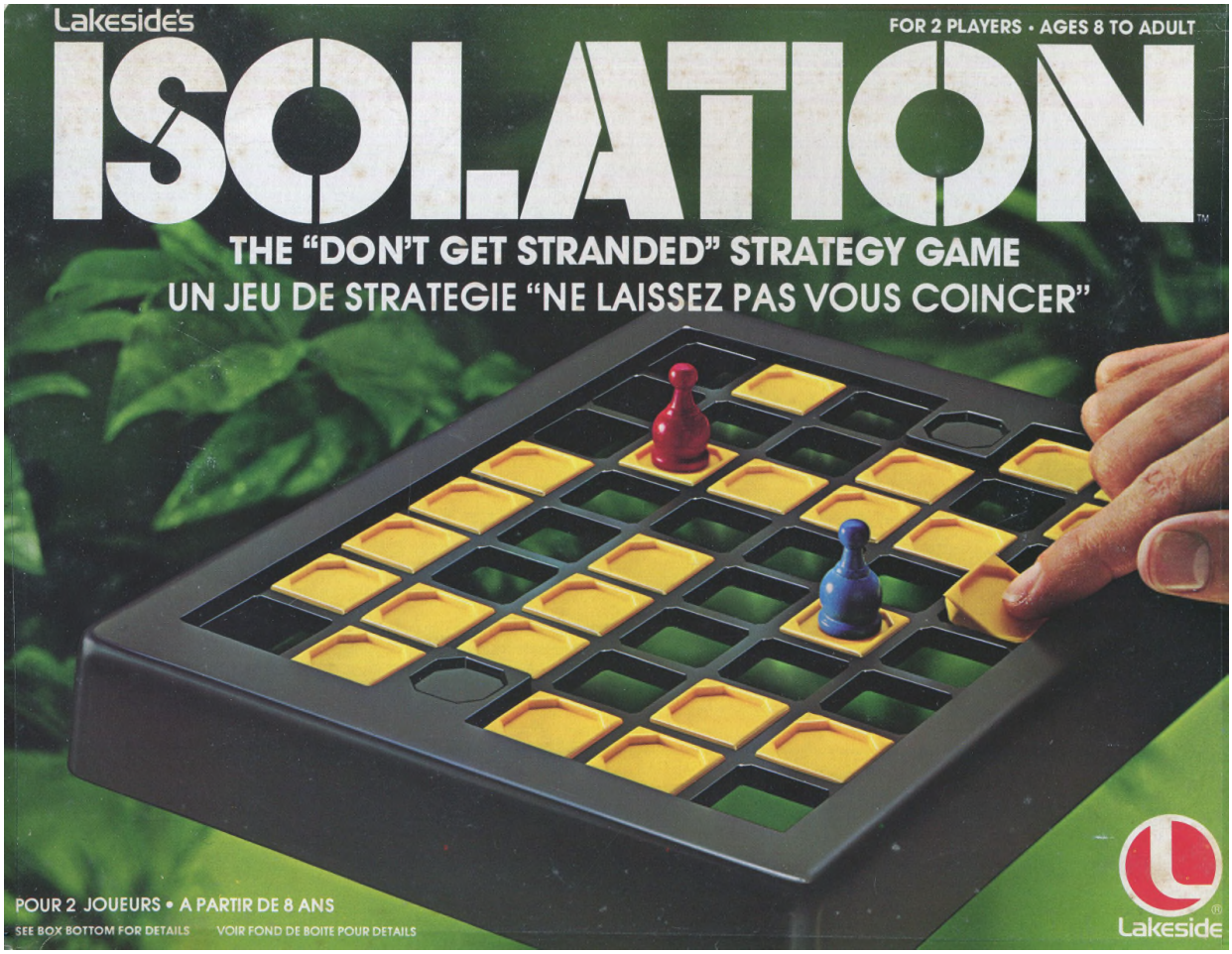
# pa1

Due Fri Oct 9 at 6:00 PM.
Submit as pa1.cs at fileupload.ca.

## Assignment

Write a C# console program to play the game of Isolation (by Lakeside). The following images show the front and back of the game box and the rule sheet.



Lakesides

FOR 2 PLAYERS • AGES 8 TO ADULT

POUR 2 JOUEURS • A PARTIR DE 8 ANS

# ISOLATION™

## THE "DON'T GET STRANDED" STRATEGY GAME

## UN JEU DE STRATEGIE "NE LAISSEZ PAS VOUS COINCER"

EACH PLAYER HAS A PAWN. THE BOARD IS COVERED WITH 46 PLAYING TILES.

CHAQUE JOUEUR A UN PION. LE TABLEAU EST COUVERT DE 46 CARREAUX A JOUER.

MOVE YOUR PAWN ONTO A TILE AND PUSH OUT ANY TILE IN FRONT OF YOUR RIVAL.

DEPLACEZ VOTRE PION SUR UN CARREAU ET FAITES SORTIR TOUT CARREAU EN FACE DE VOTRE ADVERSAIRE.

AS THE GAME GOES ON THE TILES DISAPPEAR. PLAN YOUR MOVES. DON'T GET STRANDED.

COMME LE JEU CONTINUE, LES CARREAUX DISPARAISSENT. ETUDIEZ VOS COUPS. NE VOUS LAISSEZ PAS COINCER.

TO WIN, SET THE TRAP, STOP THE LAST ESCAPE, ISOLATE YOUR RIVAL.

POUR GAGNER, METTEZ LE PIEGE, ARRETEZ LE DERNIER ECHAPPEMENT, ISOLEZ VOTRE RIVAL.

Leisure Dynamics

LEISURE DYNAMICS OF CANADA LTD., DON MILLS, ONTARIO M3A 1C6

**FULL INSTRUCTIONS INSIDE**
**INSTRUCTIONS COMPLETES A L'INTERIEUR**

Lakeside's

# ISOLATION™

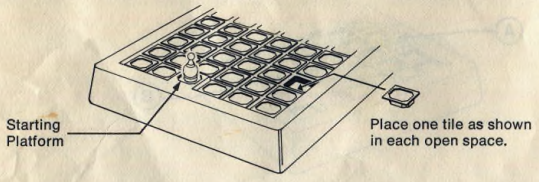
## INSTRUCTIONS:

### THE OBJECT

To isolate your opponent's piece so it is impossible for it to move.

### TO START

Cover all the squares of the playing grid with the yellow tiles. Place the pawns on the black starting platforms.

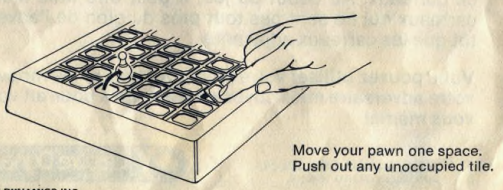



Select the starting player.

### THE PLAY

On your turn,

1. Move your pawn one space in any direction (diagonally, sideways, forward, or backward) to an adjacent tile.
2. After you move your pawn, remove one tile from anywhere on the grid by pushing it out through the board.

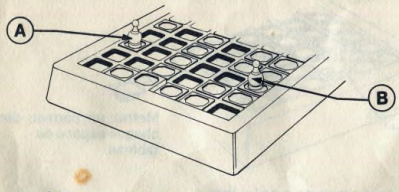





Only one pawn may occupy a tile. The starting platforms cannot be pushed out but may be used as tiled spaces throughout the game by either player.

### TO WIN

Attempt to push out the tiles so as to isolate your opponent's pawn. If your opponent cannot move, you win.




Isolate your opponent. In the diagram, player A is isolated and has lost. Player B can still move and has won the game.

### STRATEGY

Attempt to keep your opponent in areas where there are fewer tiles. In the beginning of the game, it may be useful to push out tiles which are a few grid spaces away from your opponent's pawn rather than adjacent to it.

You can use your pawn to limit your opponent's range of moves, but be careful, don't get trapped yourself.

**LEISURE DYNAMICS OF CANADA LTD.**
**DON MILLS, ONTARIO M3A 1C6**

**471015**

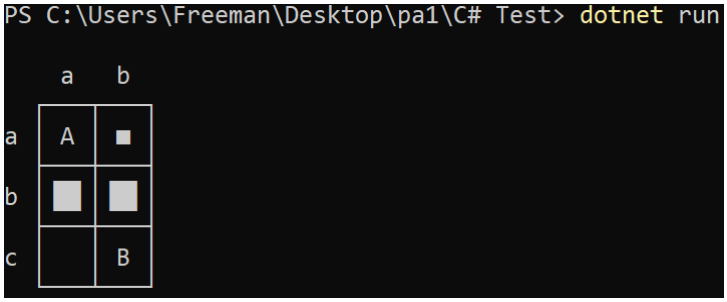
There's a slight wording conflict between the back of the box and rule sheet regarding which tiles you are allowed to remove. The back of the box says "in front of your rival" and the rule sheet says "any unoccupied tile". We will follow the rule sheet.

## How would such a game work?

Games like this which are played on the console proceed as a conversation between your program and the two users who share the console. Your program is basically a loop which runs as long as further play is possible. An iteration involves asking one of the players for their move and then showing the resulting state of the game board. Which player is being asked alternates back and forth between the two players. Before entering the loop, the program requests general information such as the player names and the starting configuration.

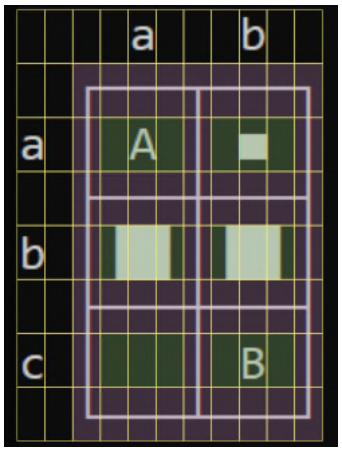
## Displaying the game board

A decent rendition of a game board like this can be done using the Unicode box-drawing characters. The provided file "box.cs" will draw a small game board as shown below.




Here, I am using "A" and "B" to indicate the tiles occupied by the pawns of players "A" and "B". The small square represents a starting platform when not occupied. The large square represents a tile which can be occupied. The empty box represents a tile which cannot be occupied because it has been removed. The letters along the sides are a coordinate system to identify tiles by their row and column. For example, the small square is located at "ab" meaning it is in row "a" and column "b".

The following image overlays a yellow grid to show each character in its own box. Highlighted in purple are the characters making up the game board tile boundaries. The tile state is always a three-character string as shown highlighted in green. The large block is made up of a right-half block followed by a full block followed by a left-half block. The others are all int the form of a space followed by a character followed by another space.



# Core representation

Use a two-dimensional array to hold the state of the tiles (removed or not). Use variables to hold which tiles are the platforms and which tiles are holding the two pawns (A and B). This means one element in the array corresponds to one three-character string in one tile on the game board. Which three-character string you display for one tile on the game board will depend on the array at that location and the variables indicating platforms and pawns. Logic in the loops to display the board can be used to add in the other characters for the tile-boundary lines and coordinate labels.

# Starting configuration

We want to ask the user for the following information: the name of player A, the name of player B, the number of rows and columns in the game board, and the positions of the two platform tiles. Allow the user to just hit enter at each prompt to get the defaults of "Player A" and "Player B" for the names and a board setup like the physical game illustrated on the box cover. Don't allow less than 4 or more than 26 rows or columns. We want the game large enough that moves are possible and small enough that the letters 'a' to 'z' can be used as coordinates.

# Game play

A player can enter their move as a four-character string like "abcd" which means they want to move their pawn to the tile in row 'a' and column 'b' then remove the tile in row 'c' and column 'd'. If both of these are possible, the move can be done by updating the array and variables appropriately. Otherwise, the user should be given a message regarding what is wrong and asked to enter the move again. A move is possible if the coordinates map to a tile on the game board, the tile is not removed, and the tile is not already occupied by a pawn. A tile removal is possible if the coordinates map to a tile on the game board, the tile is not already removed, the tile is not a platform, and the tile is not occupied by a pawn.

# How to start

I would start by creating the logic which, given the number of rows and columns, would display the game board with no platform tiles and all tiles removed. This gives the ability to see what is going on in the game for the rest of the development. I might then add the core configuration array and variables, with fixed values in place of the user input, and update the display to include the tile state (removed or not), platforms, and pawns. I would then tackle the starting configuration interaction with the user. Finally, I would code the game play loop. For a problem like this, I would also sketch a picture of the typical game board and label it by my variable names so I can keep track of their meaning.

# Questions

You may need additional information to proceed on certain parts. Please ask questions of your classmates or the teaching team. Two natural questions might be how can one get the 'a', 'b', 'c', and 'd' parts from a string "abcd" and how can one convert back and forth between the letters 'a' to 'z' which the user enters as coordinates and the numbers 0 to 25 which we use as array indices? Part of design or research is asking good questions of each other and being receptive to answers or explanations provided as you fit the pieces together into a workable solution.