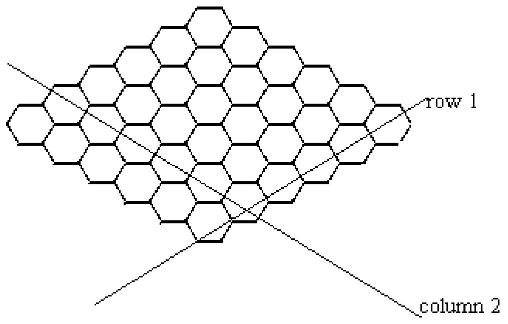
The Game of Hex

The aim of the game is for the first player to connect a hex counter owned by her on column 1 to a hex counter owned by her on column N.

Rules of Hex:

Hex is a two player strategy game played on a NxN rhombus of hexagons, as illustrated here for N=6.



1. The two players of the game are your program and the evaluation library.
2. Your program always has the first move.
3. Players alternately place hex counters on the board.
4. A hex counter may be placed at any open position on the board.
5. Two hexagons are adjacent if they share an edge.
6. Hex counters on adjacent hexagons of the same player (contestant next to contestant, or evaluator next to evaluator) are *connected.*
7. Connectivity is transitive (and commutative): if hex1 is connected to hex2 and hex2 is connected to hex3 then hex3 is connected to hex1 and hex1 is connected to hex3.

Task:

- You are required to write a program which plays the game of Hex.
- The goal of the first player (your program) is to connect a hex counter of yours on column 1 to a hex counter of yours on column N.
- The other player (evaluator's program) attempts to connect an evaluator's hex counter on row 1 to an evaluator's hex counter on row N.
- If your program plays optimally, it will always win.

Input and Output:

Your program must **not** read from or write to any files. Your program must not receive keyboard input, and must **not** produce output on the screen. It will receive all its input from the functions in the hex library. The library will produce an output file named HEX.OUT; you should ignore its contents.

At the start of the game your program will be presented with a board that may have hex counters already placed, representing a state of a game such that the first player may still win. Your program must use the functions GetMax and LookAtBoard to determine the state of the board.
At the start of the game, an equal number of hexes belongs to the evaluation program and your program.

Constraints:

1. The size of the board will always be in the range **1** to **20** inclusive.
2. Your program may have to make up to 200 moves to complete a game. The entire game must be finished within 40 seconds. It is guaranteed that the evaluation library will complete its processing within 20 seconds.

Library:

A library called *HexLib* is provided which you must link to your code. An example file, for each programming language, showing how this is done is included in the task directory. These files are TESTHEX.CPP, TESTHEX.C, TESTHEX.PAS, and TESTHEX.BAS. If you are using QuickBasic you must include the library by typing

```
QB /L HEXLIB
```

The functions in HexLib are:
(in order of Pascal, C/C++ and Basic respectively)

function LookAtBoard (row, column: integer): integer;
int LookAtBoard (int row, int column);
declare function LookAtBoard cdecl (byval x as integer, byval y as integer)
Returns
–1 if row<1 or row>N or column<1 or column>N
0 if there is no hex counter at the position
1 if the hex counter at the specified position belongs to your program (player 1)
2 if the hex counter at the specified position belongs to the
evaluation library (player 2)

procedure PutHex (row, column: integer);
void PutHex (int row, int column);
declare sub PutHex cdecl (byval x as integer, byval y as integer)
Places a contestant's hex counter at the specified row and column if the position is not occupied.

function GameIsOver: integer;
int GameIsOver (void);
declare function GameIsOver cdecl ()
Returns one of the following integers
0 the game is not over.
1 every position on the board is occupied by a hex counter.
2 your program has won.
3 the evaluation library has won.

procedure MakeLibMove;
void MakeLibMove(void);
declare sub MakeLibMove cdecl ()
Allows the evaluation library to calculate its next move and places its hex counter on the board. The change to the board will be indicated by *LookAtBoard* and the other functions.

function GetRow: integer;
int GetRow (void);
declare function GetRow cdecl ()
Returns the row of the hex counter placed by the evaluation library, or –1 if no hex counter has been placed yet. This function always returns the same value until your program calls *MakeLibMove* again.

function GetColumn: integer;
int GetColumn (void);
declare function GetColumn cdecl ()

Returns the column of the last hex counter placed by the evaluation library, or –1 if no hex counter has been placed yet. This function always returns the same value until your program calls *MakeLibMove* again.

function GetMax: integer;
int GetMax (void);
declare function GetMax cdecl ()
Returns the size of the board, N.

Scoring:

- If your program wins a game, it will score full marks for that data set.
- If your program loses a game, it will score 20% for that data set.
- If your program terminates before the end of a game or runs out of time, it will score 0 for that data set.