COMP30024 Artificial Intelligence
Christopher Leckie and Sarah M. Erfani

Project Part A: Testing the Current State

Updated By: Lida Rashidi
Last updated: 3rd March 2016

Task

The aim of this part of the project is to design and implement a Java program to test for the following queries given a partially complete board:

- Number of possible moves
- Maximum number of hexagonal cells that can be captured by one move (0, 1, 2)
- Number of hexagonal cells available for capture by a single move

This will help refresh your skills in Java, and provide you with important infrastructure that will be needed by your game playing agent in Part B. Before you read this specification, please ensure you have read “Project Specification: Rules of the Game of HexiFence”.

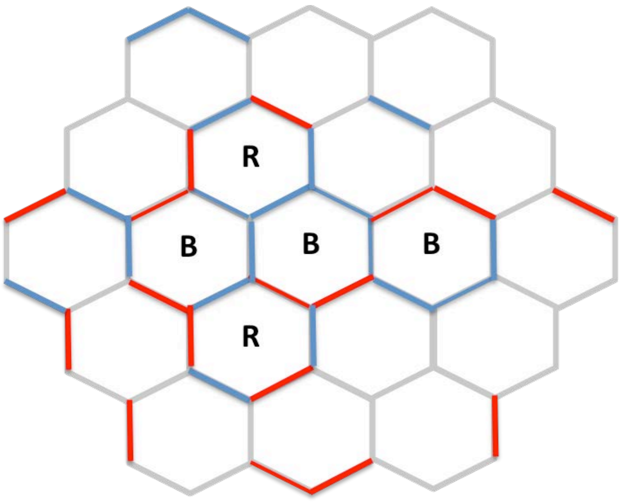
When a player makes a move, it needs to check for the queries mentioned above. Your task is to implement a program in Java 1.7 that responds to these tests for a given board configuration.

Your program should read a board configuration from the standard input.
- The first line of input contains an integer corresponding to the dimension of the board. You can assume the given dimension of the board will be *N* = *2* or *3*.
- The remaining lines of input will specify the contents of each edge in the board, one row at a time, starting from edge (0,0), i.e., the piece at (0,0), then (0,1), then (0,2), etc. The contents of the edges on the board will be encoded as one of the characters B, R, - or +, corresponding to a Blue piece, a Red piece, non-existing edge or a free edge, respectively. The contents of consecutive edges in the input will be separated by whitespace for a board dimension *N*, each remaining line of the input will have *4N-1* values.

For example, the input:

```
3
B B + + + + - - - - -
+ - + - + - + - - - -
+ + B R + B + + - - -
+ - R – B - + - + - -
R B R B B B R R + R -
+ - B – B – B – B - +
- B + R B R R B B + +
- - R - R - B - + - +
- - - + + B R + + + +
- - - - R - + - + - R
- - - - - + + R R + +
```

corresponds to the following board configuration (where R and B denote the captured cells by the red and blue player respectively):



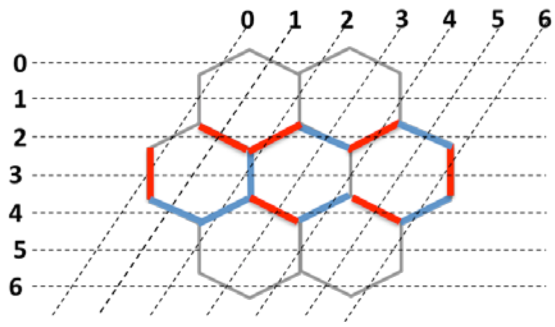
The output from your program should consist of three values where each one answers one of the mentioned queries for the given state of the board. Each of these three values should be on a separate line. The following output should be written to the standard output:

- The first line should be a value ≥ 0 corresponding to the number of possible moves.
- The second line should be a value ≥ 0 corresponding to the maximum number of hexagonal cells that can be captured by one move.
- The third line should be a value ≥ 0 corresponding to the number of hexagonal cells available for capture.

If there is a syntactic error in the input or an incorrect number of positions, your program should detect the error, output a short error message and exit. Your program should produce no other output than what is specified above.

For example, given the following board state as the input,

```
2
+  +  +  +  -  -  -
+  -  +  -  +  -  -
+  R  R  B  R  B  -
R  -  B  -  +  -  R
-  B  B  R  B  R  B
-  -  +  -  +  -  +
-  -  -  +  +  +  +
```



your program should produce the following output:

```
16
2
3
```

When Part 1 is marked, the command line input will be in a file (such as: "`java TestWin < input`").

Assessment

Part A is intended as a progress submission for the larger project. The aim of Part A is to motivate you to get started on the project. Consequently, marks will be awarded for the quality of your code, the correct operation of your program, and the efficiency of your algorithm. Your program should include comments where appropriate. You will also need to provide a pseudocode description of the algorithm you used, and discuss its time complexity.

Part A will be marked out of 8 points, and contribute 8% to your final mark for the subject. Of the 8 points, 2 points will be for the quality of your code and comments, 4 points will be for the results of testing on a set of test cases, and 2 points will be for the pseudocode description of your algorithm and the discussion of its time complexity. Note that even if you don't have a function that works correctly by the time of the deadline, you should submit anyway, since you may be awarded some points for a reasonable attempt at the project.

Questions and answers pertaining to the project will be available in the FAQ page on the LMS and will be considered as part of the specification for the project.

There should be one submission per group. You are encouraged to discuss ideas with your fellow students, but your program should be entirely the work of your group. It is not acceptable to share code between groups, nor to use the code of someone else. You should not show your code to another group, nor ask another group to look at their code. You may make use of the library of classes provided by the Russell and Norvig textbook site if you wish, provided you make appropriate acknowledgements that you have made use of this library (a copy of their library of classes will soon be provided on the MSE Student Unix servers). If your program is found to be suspiciously similar to someone else's or a third party's software, you may be subject to investigation and, if necessary, formal disciplinary action.

Please refer to http://academichonesty.unimelb.edu.au/ if you need further clarification on this point.

Submission

Please see the separate document partA-submit-2016.pdf. The submission deadline for Part A is 4.00pm Wednesday 6th April 2016.