Assignment Due Date and Time

26 November 2023 (Sun) 11:55 p.m.

Instructions to Students

1. This assignment is weighted 40% of the overall Continuous Assessment of this module.
2. This assignment is an individual assignment and should be done by you only. **Plagiarism will be treated seriously**. Any submitted assignment is found that involved wholly or partly in plagiarism (no matter the assignments are from the original authors or from the plagiarists) **will be scored Zero mark** and the students involved will be received discipline penalty set by the institute accordingly.
3. Only Java programming language is allowed to develop any required program.
4. Your programs must be well structured. A comment should be included in each class and method to state its main function. Explanation of each variable is also required. The source code must be properly indented. The first few lines in a source file must be a comment stating the name of the source file, your name, class, student number as well as the description of the purposes of the program. Marks will be deducted if any of the above mentioned comment is not included.
5. Grading of your programs will be based on correctness, quality, style, efficiency and the advanced features.
6. You are required to hand in a softcopy of the program source codes uploaded to Moodle.
7. Remember to backup all your programs.
8. Late submission will NOT be accepted.

Assignment Specification

You are asked to write a Reversi game in Java.

Information on the Reversi Game

Reversi is a board game played by two players (white and black sides) on a board with 8 rows and 8 columns. The player's goal is to have a majority of their colored pieces showing at the end of the game.

The game begins with four pieces placed in a square in the middle of the grid as shown below (Fig 1). The black player makes the first move.

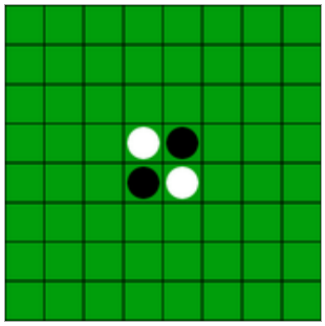


Fig 1

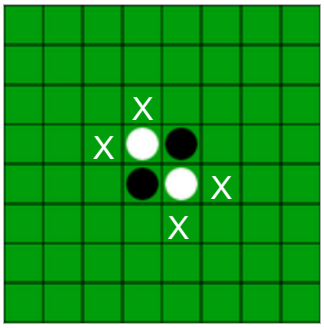


Fig 2

The player moves by placing a piece in an empty square so they can flip an opponent disc to their color **(valid move)**, i.e. there exists at least one straight (horizontal, vertical, or diagonal) occupied line between the new piece and another black piece, with one or more contiguous white pieces between them. In Fig. 2, black has the options indicated by the white X. After placing the piece, black turns over all white pieces lying on a straight line between the new piece and any other black pieces.

For example, if the black side places a piece at the position indicated in Fig 3, the result is shown at the right hand side.

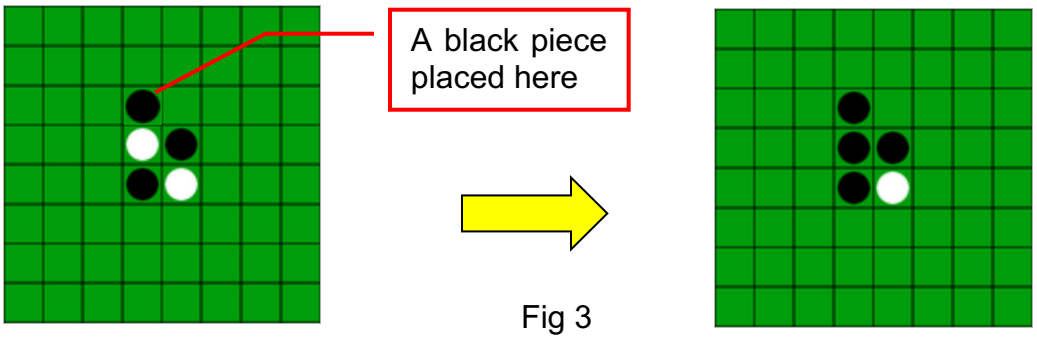


Fig 3

Players take alternate turns. If one player cannot make a valid move, play passes back to the other player. When neither player can move or the board is full, the game ends. The player with the most pieces on the board at the end of the game wins.

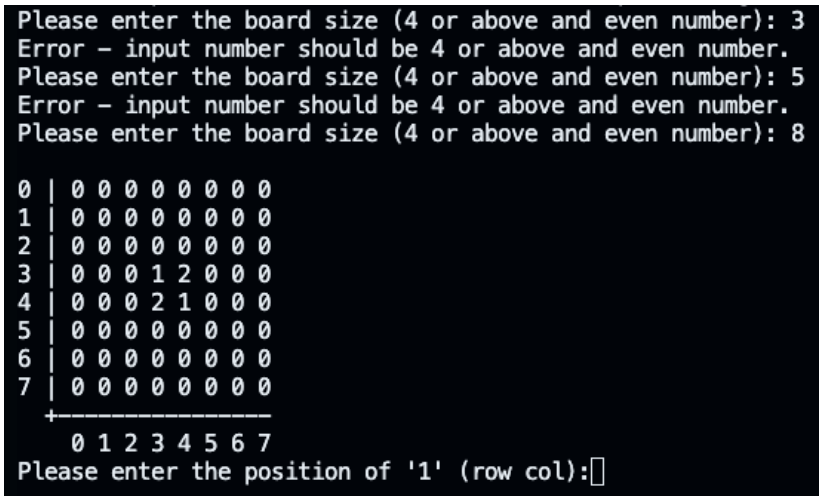
Reversi is a very popular game and it is available on many online website. To understand the game rules you can play it on

https://www.mathsisfun.com/games/reversi.html, or
http://www.webgamesonline.com/reversi/

Requirements of the Assignment

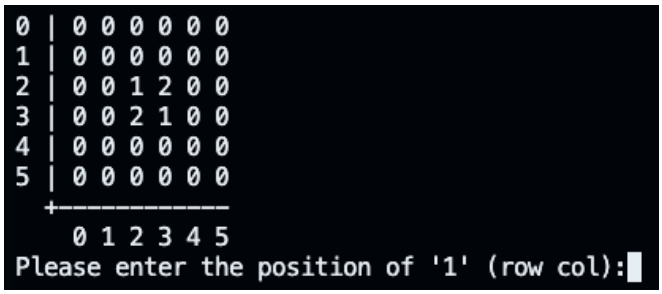
The Reversi game will be run in console mode. For simplicity, we use the letter '1' and '2' to represent the pieces of player1 (black) and player2 (white) respectively, and the letter '0' is used to represent the empty space of the grid.

You are required to create a game board with dynamic size, minimum size is 4x4 and no limit on maximum. The size should be an even number, e.g. 4x4 or 6x6 or 10x10.

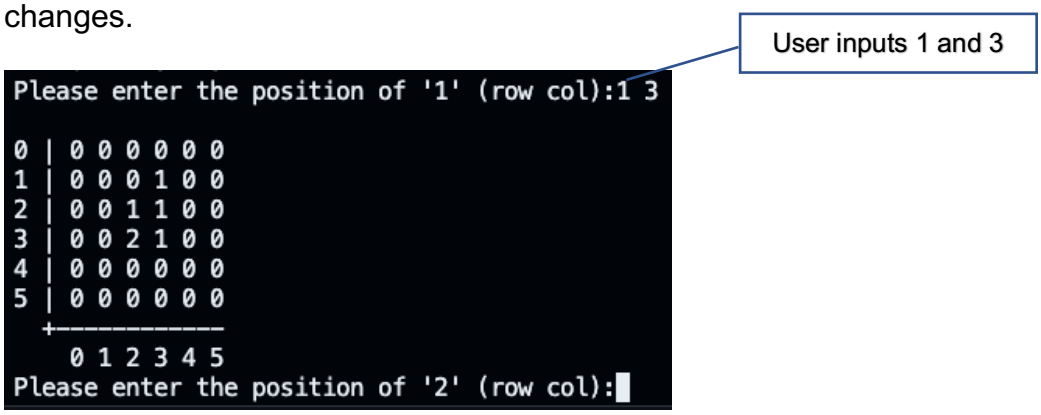


Please enter the position of '1' (row col):

To simplify the testing, this document will use 6 x 6:



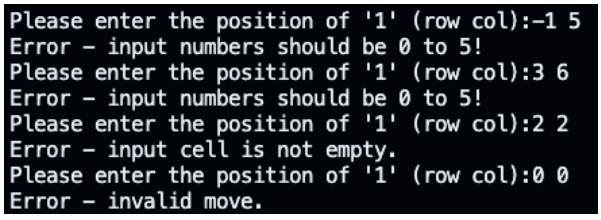
The program then asks player1 to input a pair of integers (row and column indices). The program will redraw the grid on the console with the new piece and the changes.



Then it will ask player2 to input another cell position to place its piece. After that, the program will show the updated grid on the console. It then asks player1 to input a position again. The process will be continued until neither player can move or the board is full.

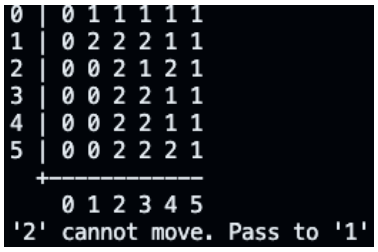
When the game ends, the program will show the result and who is the winner or a draw game.

The program should also detect the following input errors:

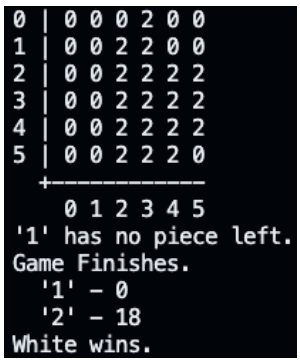


Your programme should also require to handle following problems:

If a player cannot move, pass to another player:



If a player has no more chess on the game board, she will lose the game:



Testing

You can ease the testing by using '**Copy and Paste**' rather than inputting data manually. Prepare a text file, which includes all user inputs in a game run. The following input is an example.

By using Copy and Paste, you can automatically input data in the command prompt window and then get the result automatically. **Note:** The input data will not be echoed.

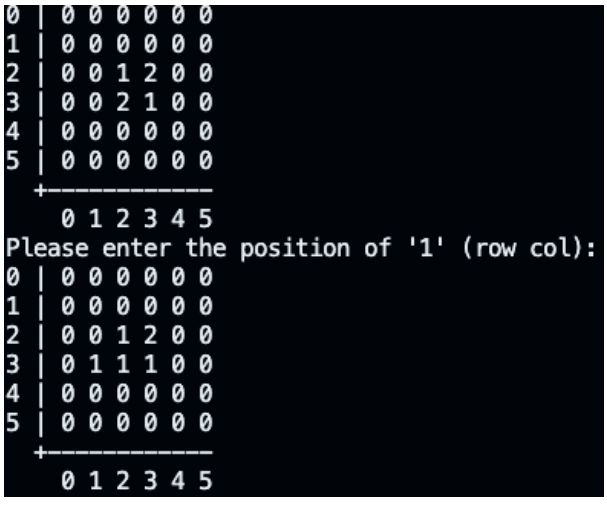
Test Case

```
6
3 1
2 1
1 3
2 4
1 4
4 2
3 4
4 3
2 0
3 0
5 2
5 3
5 1
5 0
4 0
0 3
4 4
4 1
0 2
3 5
0 4
1 2
5 4
4 5
1 1
0 1
0 0
1 5
2 5
1 0
0 5
5 5
```

Expected Output

```
Please enter the board size (4 or above and even number): 6
3 1
2 1
1 3
2 4
1 4
4 2
3 4
4 3
2 0
3 0
5 2
5 3

5 1
5 0
4 0
0 3
4 4
4 1
0 2
3 5
0 4
1 2
5 4
4 5
1 1
0 1
0 0
1 5
2 5
1 0
0 5
5 5
```




```
....... (SKIP the OUTPUT) .......

Please enter the position of '2' (row col):
```

```
Please enter the position of '2' (row col):
0 | 1 1 1 1 1 1
1 | 2 2 2 2 1 1
2 | 2 2 1 1 1 1
3 | 2 2 2 2 2 2
4 | 2 2 2 2 2 2
5 | 2 2 2 2 2 2
  +------------
    0 1 2 3 4 5
Game Finishes.
  '1' - 12
  '2' - 24
White wins.
```

Example codes for Scanner usage

Following is a **WRONG** example program to use a Scanner to do the input:

```
// create new Scanner objects in loop
do {
      Scanner sc = new Scanner( System.in);
      choice = sc.nextInt();
} while (choice != 1);
```

Following is another **WRONG** example program to use a Scanner to do the input:

```
import java.util.Scanner;
public class WrongTest {
  public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    int x;
    System.out.print("Enter x:");
    x = sc.nextInt();
  }

  public static void method2() {
    Scanner sc = new Scanner(System.in); //second Scanner objects
    int y;
    System.out.print("Enter y:");
    y = sc.nextInt();
  }
}
```

Following is an example program to use a Global Scanner to do the input.

```
import java.util.Scanner;

public class Test {
  //Global declaration for Scanner
  public static Scanner sc = new Scanner(System.in);

  public static void main(String args[]) {
    int x;
    System.out.print("Enter x:");
    x = sc.nextInt();
  }
```

```
  public static void method2() {
    int y;
    System.out.print("Enter y:");
    y = sc.nextInt();
  }

}
```

You are NOT allowed to use GUI such as JOptionPane in your program.
Following is an example program to use a Global Scanner to do the input:

```
import java.util.Scanner;

public class Test {
  //Global declaration for Scanner
  public static Scanner sc = new Scanner(System.in);

  public static void main(String args[]) {
    int x;
    System.out.print("Enter x:");
    x = sc.nextInt();
  }

  public static void method2() {
    int y;
    System.out.print("Enter y:");
    y = sc.nextInt();
  }

}
```