Sliding Blocks Puzzle in Java (60 pts.)

Program the Sliding Blocks Puzzle as described below and in your text as exercise 2, page 156. You are to implement your solution in Java. Although you will be writing your code for a total of 6 tiles, it should be scalable to other even numbers of tiles (this is a bonus).

The sliding block puzzle consists of three black tiles, three white tiles, and an empty space with an initial configuration as shown below.

```
B B B   W W W
```

The puzzle has two legal moves with associated costs:
- A tile may move to an adjacent empty location; this has a cost of 1.
- A tile can hop over one or two tiles into the empty position. This has a cost equal to the number of tiles jumped over.

The goal is to have the white tiles to the left, the black tiles to the right, and the space in the center.

You will implement the computer search for the goal in four different ways: depth first search, breadth first search, and two heuristics designed to minimize the total number of states visited. In addition to submitting your program, prepare a brief one-page report that gives the total number of states visited and the cost of the solution path for the four different searches. Explain your two heuristic evaluation functions and try to account for the search results you observed.

You can access two resources to get you started, both can be found by going to the program assignment links under cs4440 at the instructor’s website (www.cs.appstate.edu/~blk).

- There is an interactive version of the sliding blocks puzzle that you can run to get familiar with the puzzle and that you can use as source code (particularly for the GUI) in getting started on your program. You can replace the control buttons below the display to select the four search techniques and you can display the cost and number of states visited above the display.
- The code from the Bigus textbook for search algorithms is also provided; you may or may not find it useful in implementing your best first algorithm.

The major tasks you have to accomplish are:
- Write a depth first search where moves are attempted in a fixed sequence; this approach is susceptible to “going down the garden path” unsuccessfully, so order your moves wisely and set a depth limit to force some backtracking
- Write a best first search algorithm to handle breadth first search (the heuristic value always returns zero) and your two evaluation functions
- Modify the user interface as required
- After your program is working correctly, collect data and write your report
Submit your program and report using the submit program:
   u/csd/blk/bin/submit4440   p1   <files(s)>
Make sure your files are well documented, including the normal header information (name, class, program), an explanation for your heuristics, and comments as appropriate.

**BONUS #1:** (10 pts.) Make the size of the puzzle (2, 4, 6, 8, and 10 tiles) a user option. You may find some of the search techniques become excessively long for large sizes, so set a cut off value so that the search is abandoned in failure if the goal is not found after considerable searching.

**BONUS #2:** (10 pts.) Allow the user to observe the steps the computer is attempting by installing a JSlider (a class in Java) that will control the speed of execution. Make the entire program run as a thread and add a sleep time between moves. The slider will control the sleep time in msec allowing a range of 0 to 1000 msec between moves.