HOMEWORK 1

The homework has four parts. Each part must be done separately.

Write a program to experiment with the A* algorithm under various conditions. The program uses the A* algorithm to solve the sliding puzzle. The puzzle uses a n×n board. The board has numbers from 1 to n^2-1 in some random order and one zero. The zero and and an adjacent number can be swapped. The puzzle has an initial board and a goal board. The program has to find a sequence of swaps to go from the initial board to the goal board.

5 7 1	1 4 8
2 0 8	5 2 6
4 6 3	0 3 7

The program reads the puzzle information from a user specified input file. The input file has the puzzle size, the initial board, the goal board, the evaluation function option, and the heuristic function option. The evaluation function option is 1, 2 or 3. The heuristic function option is 1 or 2

Evaluation function f options:
Option 1: f = h where h is heuristic function
Option 2: f = g where g is path cost function
Option 3: f = h + g where h is heuristic function and g is path cost function

Heuristic function h options:
Option 1: h = mismatches
Option 2: h = taxi distance

The program finds a sequence of swaps to go from the initial board to the goal board. The program displays the sequence of boards on the screen. The program also displays the run time, the number of swaps, and the number of boards that were searched. The run time includes the puzzle solving time only, and excludes any time spent on file processing or other input/output activities. The run time is measured in milli seconds. Everything that is displayed on the screen is also written to a user specified output file. The outputs on the screen and the file must be formatted nicely.

Write a program that solves a sliding puzzle that works as follows. The puzzle uses a n×n board. The board has n^2 items. There are three types of items. Some items are numbers which are between 1 and n^2 inclusive. Some items are red which are denoted by R. Some items are green which are denoted by G. There is at least one red or one green. The puzzle has an initial board and a goal board. The initial board has some numbers, some reds, and some greens. They are located randomly. The goal board has all the numbers in the ascending order, followed by all the reds, followed by all the greens. The player has to find a sequence of swaps to go from the initial board to the goal board. Only two neighboring items can be swapped and they must be horizonal or

vertical neighbors. The following swaps are allowed: number and R, number and G, R and G. The following swaps are not allowed: number and number, R and R, G and G

```
2 G R        1 2 5
R 5 9        9 R R
G R 1        R G G
```

The program reads the initial board from a user specified input file. The input file has the puzzle size and the initial board. The program finds a sequence of swaps to go from the initial board to the goal board. The program displays the sequence of boards on the screen. The program also writes the sequence of boards to a user specified output file. The outputs on the screen and the file must be formatted nicely.

The program must be based on state space search, best first search algorithm or A* search algorithm, and heuristic or evaluation function. The program must clearly have search algorithm, heuristic or evaluation function, children generating function, open/closed lists, and other necessary components.

Write a program that plays the following game. The game has two players and a board. The board is n×n where n is even. The players take turns and place their pieces x and o on the board. A player may place its piece on any unoccupied location. The game ends when the board is full. The player who has the highest score wins. The score of a player is 2p + 3q where p is the number of two consecutive pieces and q is the number of three consecutive pieces. In other words, a player gets 2 points for two consecutive pieces and 3 points for three consecutive pieces. Overlaps are counted.

```
O O X O    Score of X = 2*6 + 3*2 = 18
X O X O    Score of O = 2*4 + 3*1 = 11
X X X O    X wins
O X O X
```

The program must play competitively. The program should not take more than one minute for one move for a board size 6×6 or less. The program asks the board size at the beginning. The program uses x and the human player uses o. The human player makes the first move. For each move of the human player and the program, the coordinates of the move, the board, the score of the program, and the score of the human player are displayed. The win or draw is displayed at the end of the game. Everything that is displayed on the screen is also written to a user specified output file. The outputs on the screen and the file must be formatted nicely.

The program must be based on the min max algorithm that uses depth limit, alpha beta pruning, and heuristic board evaluation. The program must clearly have min max algorithm, depth limit, alpha beta pruning, heuristic evaluation function, and other necessary components.

Write a program that solves a puzzle that works as follows. It uses a $n^2 \times n^2$ board. The numbers 1, 2, 3 . . . n^2 are used to fill the slots. There are five types of slots. Some slots have initial numbers and the program cannot change them. These slots are denoted by their numbers. Some slots are blacked out and the program cannot fill them. These slots are denoted by b. Some slots can only have odd numbers 1, 3, 5, 7 . . . These slots are denoted by o. Some slots can only have even numbers 2, 4, 6, 8 . . . These slots are denoted by e. The remaining slots can have any number. These slots are denoted by w. The program fills the w, o, e slots. The program does not fill the b slots and the slots with the initial numbers. Each row must have distinct numbers. Each column must have distinct numbers. Each n×n region must have distinct numbers. There are n^2 number of n×n regions.

```
w  w  b  b  w  e  6  o  w
w  2  w  w  w  w  w  w  e
w  w  w  5  8  w  w  b  w
7  w  o  w  w  w  e  w  w
w  b  w  w  9  w  w  w  w
w  w  w  w  e  w  w  w  w
3  o  w  b  w  w  9  w  w
w  w  w  w  w  e  w  6  w
w  w  4  w  b  w  w  o  w
```

The program reads the puzzle from a user specified input file. The input file has the puzzle size and the puzzle itself. The program solves the puzzle. The program displays the puzzle and its solution on the screen. The program also writes them to a user specified output file. The outputs on the screen and the file must be formatted nicely.

The program must be based on the constraint satisfication algorithm that uses recursion, backtracking, and partial solution checking. The program must clearly have constraint satisfication, recursion, backtracking, partial solution checking, and other necessary components.

Program must be written Java language. Write comments explaining all steps. Run the program on the data provided separately. Submit the program and the results.