CS 4613 **Fall 2020**
Project 1: 11-Puzzle Problem **E. K. Wong**

Total # points = 100. Figures 1 and 2 are on page 2.

Project Description: Implement the *A** algorithm with *graph search* for solving the 11-puzzle problem as described below. Use *sum of Manhattan distances of tiles from their goal positions* as heuristic function. [Reminder: *graph search* does not allow repeated states.]

11-puzzle problem: On a 3 x 4 board there are 11 tiles numbered from 1 to 11 and a blank position. A tile can slide into the blank position if it is horizontally or vertically adjacent to the blank position. Given a start board configuration and a goal board configuration, find a move sequence with a minimum number of moves to reach the goal configuration from the start configuration. (Note: the 3 x 4 board has 3 rows and 4 columns.)

You can work on this project by yourself or form a team of two students to work on the project. You can discuss with your classmates or other teams on how to do the project but everyone or team is expected to write their own code and submit their own program.

Input and output file format: Your program will read in the initial and goal states from a text file that contains 7 lines as shown in Figure 1 below. Lines 1 to 3 contain the tile pattern for the initial state and lines 5 to 7 contain the tile pattern for the goal state. Line 4 is a blank line. *n* and *m* are integers that range from 0 to 11. Integer 0 represents the blank position and integers 1 to 11 represent tile numbers. Your program will produce an output text file that contains 12 lines as shown in Figure 2 below. Lines 1 to 3 and lines 5 to 7 contain the tile patterns for the initial and goal states as given in the input file. Lines 4 and 8 are blank lines. Line 9 is the depth level *d* of the shallowest goal node as found by your search algorithm (assume the root node is at level 0.) Line 10 is the total number of nodes *N* generated in your tree (including the root node.) Line 11 contains the solution that you have found. The solution is a sequence of actions (from root node to goal node) represented by the A's in line 11, separated by blanks. Each A is a character from the set {L, R, U, D}, representing the left, right, up and down movements of the blank position. Line 12 contains the $f(n)$ values of the nodes along the solution path from the root node to the goal node, separated by blanks. There should be *d* number of A values in line 11 and $d+1$ number of *f* values in line 12.

Testing your program: Four input test files will be provided on NYU Classes for you to test your program. You can assume that all four test files are solvable; i.e., they have solutions.

Recommended languages: Python, C++/C and Java. If you would like to use a language other than these three, send me an email first.

Submit on NYU Classes by due date:

1. A text file that contains the source code. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have comments in your source code.
2. The output text files generated by your program for test input files 1 to 4.
3. A PDF file that contains instructions on how to run your program. If your program requires compilation, instructions on how to compile your program should also be provided. Also, copy and paste the output text files and your source code onto the PDF file (to make it easier for us to grade your project.) This is in addition to the source code file and output text files that you have to hand in, as described in (1) and (2) above.

n n n n
n n n n
n n n n

m m m m
m m m m
m m m m

Figure 1. Input file with 9 lines.

****
n n n n
n n n n
n n n n

m m m m
m m m m
m m m m

d
N
A A A A A A A ……
f f f f f f f f ……..

Figure 2. Output file with 12 lines.