CS380 HW Assignment 2

Due date: see syllabus

Programming Assignment (100 points)
Using the code you wrote for assignment 1, write a function that solves a given sliding bricks puzzle using A*.

A. As a heuristic, use the Manhattan distance between the master brick and the goal. Notice that the search space is a graph, so you will have to keep track of all the states visited so far, and make sure your algorithm does not get stuck in loops.

B. Devise a second better ADMISSIBLE heuristic. Notice that the performance of A* strongly depends on the heuristic you use. Try to design the best heuristic you can, and turn in an explanation of the heuristic in a README text file (please recall that we do NOT accept Microsoft Word files, use plain text files).

When the solution is found, it should be printed to the screen. Print the list of moves required to solve the state, and the final state of the puzzle, for example: (**pay attention to spaces and newlines**)

```
(2,left)
(4,down)
(3,right)
(2,up)
(2,up)
5,4,
1,2,2,1,1,
1,0,0,3,1,
1,0,0,4,1,
1,1,1,1,1,
```

Main Function
Write a main function that, when the program is run, calls the function(s) you wrote above in order to accomplish the following:

1. Load file SBP-level1.txt from the current directory and display to the screen the solution obtained using A* with the Manhattan distance heuristic in the format specified above, followed by a line of the form:

#nodes time length

   where *#nodes* is the number of nodes explored, *time* is the time the search took in seconds and fractions of seconds (e.g., 2.53 for 2 seconds and 53/100) and *length* is the length of the solution found.
   Before the output of solution and statistics, display a line of the form:

level-name heuristic

where *level-name* is the name of the file you loaded (e.g., SBP-level1.txt) and heuristic is either the string *manhattan* or the string *custom*.

2. Display to the screen the solution obtained using A* with your custom heuristic in the format specified at step 1.
3. Repeat steps 1 and 2 for file SBP-level2.txt.

IMPORTANT: write all of the code above to be run from command line and to display its output to the console. Do not create any graphical user interfaces and do not ask the user for any input.

To help you with testing, additional files have been provided.

What to Submit

All homework for this course must be submitted using Bb Learn. Do not e-mail your assignment to a TA or Instructor. If you are having difficulty with your Bb Learn account, you are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment is due. If you have any doubts, complete your work early so that a TA or someone from IRT can help you if you have difficulty.

For this assignment, you must submit:

- Your C/C++ source code, and written documentation for your program, including compilation and execution instructions.
    - The source code must include a Makefile file written for use on tux.
    - Running “make” must produce an executable named “hw2” in the directory where Makefile is. “hw2” must implement the main function and associated functions/methods described above.
- An executable called “hw2” that runs on tux.
- A plain text file called “output-hw2.txt” and **generated on tux (very important!)** which shows the output your program generates when run. You can easily generate this file using redirection, e.g.: “./hw2 > output-hw2.txt”.
- Use a compression utility to compress your files into a single file (with a .zip extension) and upload it to the assignment page.

Academic Honesty

You must compose all program and written material yourself. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a TA during office hours.