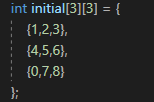
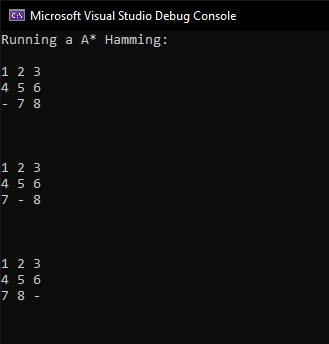
**Programming Assignment 1 Report Dimitrije Prosevski**

**Introduction:**

For the simplicity of fast testing and shortness of the output I used easy cases below. I did not try to implement random cases because of time shortage. Program will offer to input your elements for each slot of the puzzle upon the start.

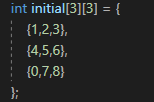
**Result for A\* Hamming:**

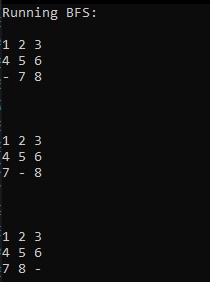
 Initial input:

 Output:

A\* successfully implemented, every node generated is put into the queue. The node we pop from the queue is the lowest cost node. Once we pop the node that has the same state of the board as goal state board, the search is done. So, the goal state to output will be the best possible path to the goal state found in the whole tree. A\* is working slower than BFS but with much less memory required.

**Result for BFS:**

 Initial input:

 Output:

BFS is implemented in a way where in each level of the tree it is scanning through the nodes to find all the possible children of each node of the level. Once it encounters the first goal state it will stop the code and prompt the result. BFS works a lot faster than A\* in this case, but memory requirement is much increased.

To test the program, go to **dprosev** on tesla server, run command

**cd AI387/proj1**

**make**

**./RunCode**

To change the search algorithm the variable “runAstar” on line 22 needs to be changed



If bool runAstar = true, A\* will be ran.

If runAstar = false, BFS will be ran.