CS 540-1: Introduction to Artificial Intelligence
Homework Assignment # 3
Assigned: 9/29
Due: 10/6 before class

Hand in your homework:

This homework includes only a programming portion. The solution to the programming problem should be coded in Java, and **you are required to use only built-in libraries** to complete this homework. Please submit a source code file named **Search.java** with **no package statements**, and make sure your program is runnable from command line.
Go to Learn@UW My Course Dashboard, choose the 540 course, choose Assignments/Dropbox, click on hw3: this is where you submit your file.

Late Policy:

All assignments are due at the beginning of class on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday 9:30 a.m., and it is handed in between Wednesday 9:30 a.m. and Thursday 9:30 a.m., 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline. A total of two (2) free late days may be used throughout the semester without penalty.

Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Collaboration Policy:

You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas with classmates, TAs, and instructor in order to help you answer the questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers or code fragments from anyone or anywhere
- not to allow your answers to be copied
- not to get any code on the Web

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we suggest that you specifically record on the assignment the names of the people you were in discussion with.

Problem: Search for General Two Water Jugs Problem [100 points]

The goal of this assignment is to become familiar with state space search algorithms and apply them to real-world problems.
You are given 2 water jugs with capacity m, n liters (m, n are positive integer numbers and m $\geq$ n). You have to use these two jugs to measure d liters (d is a positive integer number) where d $\leq$ m. The operations you can perform are:

- Empty a jug
- Fill a jug
- Pour water from one jug to the other until one of the jugs is either empty or full

Use Breadth First Search and Depth First Search, separately, to solve the water jug problem. Your algorithms should maintain a CLOSED data structure to avoid all states that have already been expanded. Write a program **Search.java** that does the following.

- Input: 3 positive integers m, n, d where m $\geq$ n, d (see examples below for input format)
- Output: Algorithm should be run in the order of **BFS, DFS**. Print the algorithm you use. Each search algorithm should then output an iteration part and a result part.
  - Iteration part: print "Iteration :" in the first line. Then print 3 lines for every iteration as follows.
    - Iteration number (starts from 0);
    - The states in the CLOSED structure (in any order).
    - The successors in the order you put into the OPEN data structure in this iteration. When expanding a node, make sure multiple successors are placed in the OPEN data structure in this order: "(1 first) m fill > (2) n fill > (3) m empty > (4) n empty > (5) m pour > (6 last) n pour". If there are no successors to add to OPEN in an iteration, print an empty line.
  - Result part: print "Result :" in the first line. Then print the solution path found. In the case the problem is unsolvable, print only "Unsolvable" instead.

Each state of two jugs is represented as a pair (x, y) where x is the amount of water in the m-liter jug, and y is that of the n-liter one. The solution path is a sequence of states (x, y) separated by one space. The initial state is **(0, 0)**, and the goal state is defined to be **(d, 0)**.

Here is a partial example of running the program from the command line:
$ *java Search* 3 2 2
BFS
Iteration :
0
(0, 0)
(3, 0) (0, 2)
1
(0, 0) (3, 0)
(3, 2) (1, 2)
2
(0, 0) (3, 0) (0, 2)
(2, 0)
Result :

(0, 0) (0, 2) (2, 0)
DFS
$Iteration$:
...
$Result$:
...