HW#1.1 (Programming)

Start Assignment

- Due Friday by 11:59pm
- Points 100
- Submitting a file upload
- File Types zip
- Available Jan 29 at 9am - Feb 14 at 11:59pm
- This is a group assignment. Form a group of two
  - If there are odd number of you in class, a group of one or three could be formed .
- Please note that
  - Although this assignment is a group effort, each team member must understand the codes, write their own version, and submit it individually in Canvas, and must specify the full name of each teammate.
  - While some similarities between teammates' code are acceptable to a certain extent, any evidence of copy-pasting or submitting code without demonstrating a clear understanding of the logic will not be accepted and will result in a zero grade.
- Write a program (in Java only) to solve the following puzzle.
  - Given 3 water jugs with the specified capacities and initial amount water, print out the state transition graph as shown below.
  - You need to indicate if there is any state that can be produced from multiple (n>1) parent states by adding a notation such as p2, p3, ...
    - In the figure below, notation p2 indicates the current state could be reached (produced) from two (n=2) different parent states.
- **Grading**:
  - Each instance weights 20 points.
  - Your program should be named as lastName1_lastName2.java and should take 6 command line arguments where the first 3 specify he capacities of jug A, B, and C and the rest the initial amount of water in these jugs.
    - Example:
      - javac lastName1_lastName2.java
      - java lastName1_lastName2.class 3,5,11, 0,2,11
      - where (3,5,1) are the capacities of jug A, B, and C, and (0,2,11) are the initial amount of water in these jugs.
  - Your program may be tested with **different** instances. For example,
    - Instance 1 may be changed as follows
      - **capacity: {'Jug A': 6, 'Jug B': 10, 'Jug C': 16}**

        - **Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 16}.**
    - Instance 2 may be changed to the example used in pop quiz #1.2 as follow
        - **capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 11}.**
        - **Initial amount of water: {'Jug A': 0, 'Jug B': 2, 'Jug C': 11}.**
  - One point is deducted for each incorrect number at any step.
  - Up to five points may be deducted for improper format of the program outputs.
    - The order of new measurements at each step should be in ascending order.
    - **The new measurements at each step do not have to be printed out in the center of the 2nd column.**
    - The order of new states at each step does not matter.

- **Submission**:
  - Compress your program source code and screen shots of your program outputs.
  - Submit the .zip file.

- **Expected program outputs:**
  - (Note that notations p2 below mean that the state can be produced from 2 parent states.)

```
Instance 1:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 13}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 13}.
Step: New Measurements:  New States
----:-----------------:----------------------------------------
   0:       0, 13      : [0, 0, 13]
   1:    3, 5, 8, 10   : [3, 0, 10], [0, 5, 8]
   2:         2        : [0, 3, 10], [3, 5, 5]p2, [3, 2, 8]
   3:       7, 11      : [3, 3, 7], [0, 2, 11]
   4:         1        : [1, 5, 7], [2, 0, 11]
   5:       6, 12      : [1, 0, 12], [2, 5, 6]
   6:         4        : [0, 1, 12], [3, 4, 6]
   7:         9        : [3, 1, 9], [0, 4, 9]

Instance 2:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 11}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 11}.
Step: New Measurements:  New States
----:-----------------:----------------------------------------
   0:       0, 11      : [0, 0, 11]
   1:     3, 5, 6, 8   : [3, 0, 8], [0, 5, 6]
   2:         2        : [0, 3, 8], [3, 5, 3]p2, [3, 2, 6]
   3:         9        : [3, 3, 5], [0, 2, 9]
   4:         1        : [1, 5, 5], [2, 0, 9]
   5:       4, 10      : [1, 0, 10], [2, 5, 4]
   6:                  : [0, 1, 10], [3, 4, 4]
   7:         7        : [3, 1, 7], [0, 4, 7]

Instance 3:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 17}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 17}.
Step: New Measurements:  New States
----:-----------------:----------------------------------------
   0:       0, 17      : [0, 0, 17]
   1:    3, 5, 12, 14  : [3, 0, 14], [0, 5, 12]
   2:        2, 9      : [0, 3, 14], [3, 5, 9]p2, [3, 2, 12]
   3:       11, 15     : [3, 3, 11], [0, 2, 15]
   4:         1        : [1, 5, 11], [2, 0, 15]
   5:       10, 16     : [1, 0, 16], [2, 5, 10]
   6:         4        : [0, 1, 16], [3, 4, 10]
   7:         13       : [3, 1, 13], [0, 4, 13]

Instance 4:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 19}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 19}.
Step: New Measurements:  New States
----:-----------------:----------------------------------------
   0:       0, 19      : [0, 0, 19]
   1:    3, 5, 14, 16  : [3, 0, 16], [0, 5, 14]
```

```
   1:   3, 5, 14, 16  : [3, 0, 16], [0, 5, 14]
   2:      2, 11      : [0, 3, 16], [3, 5, 11]p2, [3, 2, 14]
   3:      13, 17     : [3, 3, 13], [0, 2, 17]
   4:        1        : [1, 5, 13], [2, 0, 17]
   5:      12, 18     : [1, 0, 18], [2, 5, 12]
   6:        4        : [0, 1, 18], [3, 4, 12]
   7:        15       : [3, 1, 15], [0, 4, 15]

Instance 5:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 23}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 23}.
Step: New Measurements:  New States
----:----------------:----------------------------------------
   0:      0, 23      : [0, 0, 23]
   1:   3, 5, 18, 20  : [3, 0, 20], [0, 5, 18]
   2:      2, 15      : [0, 3, 20], [3, 5, 15]p2, [3, 2, 18]
   3:      17, 21     : [3, 3, 17], [0, 2, 21]
   4:        1        : [1, 5, 17], [2, 0, 21]
   5:      16, 22     : [1, 0, 22], [2, 5, 16]
   6:        4        : [0, 1, 22], [3, 4, 16]
   7:        19       : [3, 1, 19], [0, 4, 19]
```

```
Instance 4:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 19}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 19}.
Step: New Measurements:  New States
----:----------------:----------------------------------------
   0:      0, 19      : [0, 0, 19]
   1:  16, 3, 5, 14   : [3, 0, 16], [0, 5, 14]
   2:      2, 11      : [0, 3, 16], [3, 5, 11]p2 , [3, 2, 14]
   3:     17, 13      : [3, 3, 13], [0, 2, 17]
   4:        1        : [1, 5, 13], [2, 0, 17]
   5:     18, 12      : [1, 0, 18], [2, 5, 12]
   6:        4        : [0, 1, 18], [3, 4, 12]
   7:       15        : [3, 1, 15], [0, 4, 15]
----:----------------:----------------------------------------
Instance 5:

capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 23}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 23}.
Step: New Measurements:  New States
----:----------------:----------------------------------------
   0:      0, 23      : [0, 0, 23]
   1:  18, 3, 20, 5   : [3, 0, 20], [0, 5, 18]
   2:      2, 15      : [0, 3, 20], [3, 5, 15]p2 , [3, 2, 18]
   3:     17, 21      : [3, 3, 17], [0, 2, 21]
   4:        1        : [1, 5, 17], [2, 0, 21]
   5:     16, 22      : [1, 0, 22], [2, 5, 16]
```

```
   6:         4         : [0, 1, 22], [3, 4, 16]
   7:        19         : [3, 1, 19], [0, 4, 19]
   ----------------------------------------------
```

Hints for implementation of pouring water from one jug to another jug.

```
class State {
    public ArrayList<Integer> m_water;
    int m_level ;
    ArrayList<State> m_parents;

    ...
}

or even better with the inheritance:
```

```
class State extends ArrayList<Integer> {
    int m_level ;
    ArrayList<State> m_parents;
    public State (ArrayList<Integer> water, int level) {
        super (water);
        m_level = level;
        m_parents = new ArrayList<> ();
    }
    public int getLevel () { return m_level ; }
    public void addParent ( State par ) {
      m_parents.add (par);
    }
    public ArrayList<State> getParents () {
      return m_parents;
    }
    @Override
    public String toString () {
      String ans = "" + super.toString() ;
      int n = m_parents.size();
      if ( n>1 ) {
        ans += "p" + n;
      }
      return ans;
    }
}
```

```
/**
 * Produce new states from the given state current
 * @param current The current state
 * @return void
 */
 public  void produce (State current) {
     ArrayList<Integer> water = current.getWater();
     int level = current.getLevel();
     int n = water.size();     // number of water jugs
     for (int i=0; i < n; ++i ) {
       for (int j=0; j < n; ++j ) {
           ArrayList<Integer> water = current.getWater();
           int x = Math.min ( water.get(i), m_cap.get(j)-water.get(j));
           if (i!=j && x > 0 ) pourWater (current, i, j);
        }
     }
 }
```

Hints for producing all states.

Below is an illustration that shows how to generate all the states using an ArrayList as a FIFO Queue (remove-at-front and insert-at-rear).

```
capacity: {'Jug A': 3, 'Jug B': 5, 'Jug C': 11}.
Initial amount of water: {'Jug A': 0, 'Jug B': 0, 'Jug C': 11}.
Step: New Measurements:  New States
----:-----------------:-------------  -------------------------
   0:      0, 11      : A=[0, 0, 11]  Notation: ^ = pointer to the current state in the arrayList,
                                                * = new states produced by the current state

                                                    arrayList: A,
                                                               ^
   1:   8, 3, 5, 6    : B=[3, 0, 8], C=[0, 5, 6]
                                                    arrayList: A,B,C
                                                               ^ * *
   2:        2        : D=[0, 3, 8], E=[3, 5, 3]p2 , F=[3, 2, 6]
                                                    arrayList: A,B,C,D,E
                                                                 ^   * *
                                                    arrayList: A,B,C,D,E,F
                                                                   ^     *
   3:        9        : G=[3, 3, 5], H=[0, 2, 9]
                                                    arrayList: A,B,C,D,E,F,G
                                                                     ^     *
                                                    arrayList: A,B,C,D,E,F,G
                                                                       ^
                                                    arrayList: A,B,C,D,E,F,G,H
                                                                       ^   *
   4:        1        : I=[1, 5, 5], J=[2, 0, 9]
                                                    arrayList: A,B,C,D,E,F,G,H,I
                                                                         ^   *
                                                    arrayList: A,B,C,D,E,F,G,H,I,J
                                                                           ^   *
   5:      10, 4      : K=[1, 0, 10], L=[2, 5, 4]
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K
                                                                           ^   *
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L
                                                                             ^   *
   6:                 : M=[0, 1, 10], N=[3, 4, 4]
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M
                                                                             ^   *
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N
                                                                               ^   *
   7:        7        : O=[3, 1, 7], P=[0, 4, 7]
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N,O
                                                                               ^   *
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P
                                                                               ^   *
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N,Q,P
                                                                                 ^
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N,Q,P
                                                                                   ^
                                                    arrayList: A,B,C,D,E,F,G,H,I,J,K,L,M,N,Q,P
                                                                                     ^ <--- queue empty
```