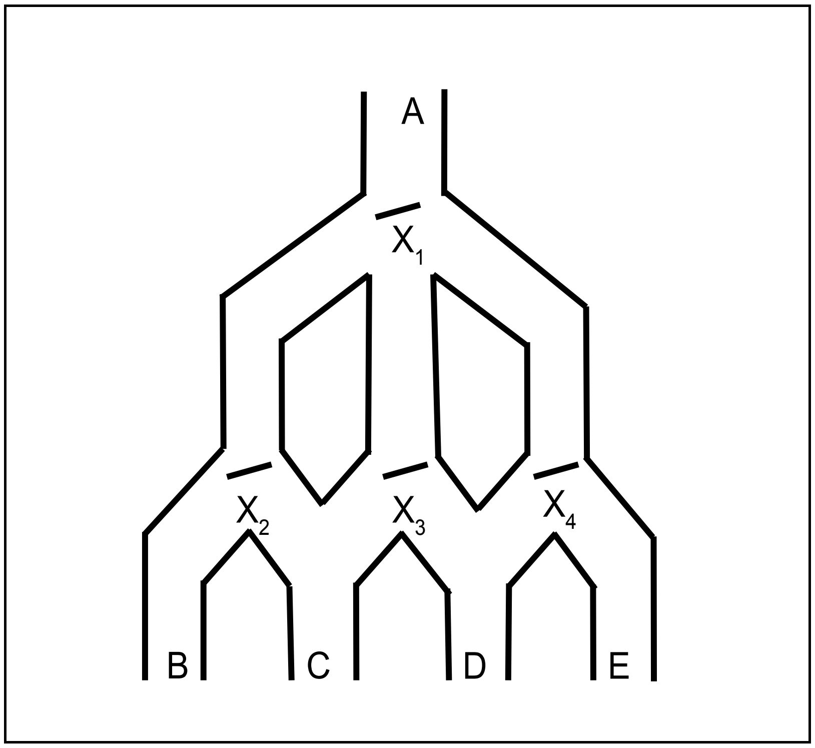
Due 10/2

For this program you have two options for what you are going to do. The first is to construct a DFA using JFLAP that simulates the following game (modified problem 2.3 from Hopcroft/Ullman). We drop a marble down the chute labeled A. If the last marble that we drop down the chute comes out one of the chutes labeled B or D, then we win. If the last marble that we drop down the chute comes out one of the chutes labeled C or E, then we loose. The second option is to simulate the operation of the game with an application program, C, C++, or Java. In this case, your program will take two inputs, the first being the gate directions and the second being the input string to simulate and your program will output the sequence of gate configurations and the chute that the last marble exits.

To simulate this we have an input alphabet of {0, 1}. An input symbol of '0' or '1' simulates dropping a marble down chute A, but they interact differently with gate X_1. Each time a marble touches a gate it causes the gate to toggle. For gates X_2, X_3, and X_4, they toggle from right to left and left to right. But gate X_1 has three possible directions – left, center, or right. On an input of 0, X_1 toggles from left to right to center to left (and so on). On an input of 1, X_1 toggles from left to center to right to left (and so on). When a marble touches a gate, the gate toggles and the marble goes in the direction that the gate was set to prior to being toggled. Initially all of the gates are configured to the left (as shown in the diagram). As an example, if all of the gates are positioned to the left, then dropping a 0 marble down chute A results in the marble exiting gate B and toggling gate X_1 to the right and gate X_2 to the right. If all of the gates are positioned to the left, then dropping a 1 marble down chute A results in the marble exiting gate B and toggling gate X_1 to the center and gate X_2 to the right.



Due 10/2

If you construct a DFA using JFLAP that simulates this marble game, the DFA should accept those strings that result in winning and reject those strings that result in loosing. There is a maximum of 48 states (24 accept, 24 reject), but only 36 are reachble from the start state, 18 accept states and 18 reject states.

For a JFLAP submisions, e-mail the JFLAP file to me (david.garrison@binghamton.edu) by 11:59:59.999pm on the date due. The filename is to be your last name in lower case followed by "_p2.jff" (my filename would be "garrison_p2.jff").

For application program submissions, e-mail your program (source file(s) and makefile if required) to me (david.garrison@binghamton.edu) by 11:59:59pm on the date due. Your main executable filename is to be your last name (lower case) followed by "_p2". For example, my executable filename would be "garrison_p2.class" for Java after compiling "garrison_p2.java" or it would be "garrison_p2" after compiling my C or C++.

The subject of your e-mail is "CS 373 program 2".

Assuming that I got my version of the program correct, the following may be helpful.

Trace of configurations for various inputs (values for $X_1X_2X_3X_4$ and string to process).

```
LLLL 0000
LLLL->RRLL->CRLR->LRRR->RLRR C

LLLL 1111
LLLL->CRLL->RRRL->LRRR->CLRR C

LLLL 01010101
LLLL->RRLL->LRLR->RLLR->LLLL->RRLL->LRLR->RLLR->LLLL E

LLLL 10101010
LLLL->CRLL->LRRL->CLRL->LLLL->CRLL->LRRL->CLRL->LLLL D
```

For what this is worth, I did both the application program (in Java) and the JFLAP DFA. I wrote the application program, and then added more code to have my application write the JFF (since there is no way that I could draw the JFLAP DFA without any mistakes).

Your grade will be based on the percentage of my test strings that your program gets the correct result on.