**Grading and expectations:**

* I will run your program giving it the filename as a command line argument:
  + ./yourCoolProgram testfile.txt
  + If you don’t know how to read a filename from the command line argument:
    - C++:
      * Instead of using int main(void) your main declaration will be

int main(int argc, char\* argv[])

* The filename you’ll open is argv[1]
* See also google searches; Here’s one of the first results I found typing ‘argv cpp’ <https://stackoverflow.com/questions/3024197/what-does-int-argc-char-argv-mean>
* here’s a stack overflow question whose answer has a working example for opening a file (rather than reading from cin <https://stackoverflow.com/questions/5353466/how-do-i-read-input-from-files>
  + - Python
      * As above, but *import sys*, and use *sys.argv[1]*
      * Or use optparse or optargs, depending on python version
    - Java
      * Your basic main static member is:

public static void main (String[] args)

* + - * IIRC, args[0] is the first argument, not the application name (contrast with c++ and other c-derived languages)
* I will be piping the output of your program to my own grading program. It will read your output and compare it to the calibrated correct output. I will be trimming whitespace, however it must be correctly comma and bracket delimited.
* If your output matches the expected output, you get full credit (yay!)
* If your output doesn’t match the expected output, it’s considered wrong and I send it back to you with the test file. It is up to you to fix the program and resubmit it for one less point.
* Each time I send a program back, or each day it is late, you lose one point. (For example, if it’s late two days and I send it back once before it’s corrected, that’s 7/10)
* You are required to meet the expected input / output standards above. If your program doesn’t *(EVEN IF IT CORRECTLY RUNS YOUR TEST DATA)*, it is still considered incorrect and must be resubmitted.
* I run Linux. I accept:
  + gcc, g++
  + Python2 and python3
  + Java

Chapter 9:

Page 638. Choose one of 10, 11, 13, 14.

Input:

- A plaintext file of exactly two lines

- The first line completely describes a finite set.

- The second line contains a 0-1 matrix in the format { {row}, {row}, ... , {row} }

- You can assume that if you have *n* elements on the first row, that the matrix will be a well-defined *n* x *n* matrix.

Example:

a, b, c, d, e, f, g, h, i,

{ {0, 1, 1, 0, 1, 0, 1, 1, 1}, {0, 1, 1, 1, 1, 1, 1, 0, 0},  [there are 7 more of these for a total of 9 rows] }

You can find an example input file here: <http://www.math.wichita.edu/~hammond/TestData.txt>

Output:

Output

- To stdout: An *n x n*  0-1 matrix in the formatted as above { {row}, {row}, ... , {row} }

**Chapter 10:**

Pick one of question #10 or #13 on page 742. The input/output depends on the problem chosen.

If you choose #10:

* Input:
  + Plaintext file with four lines
  + The first and third lines list the vertices of a graph
  + The second and fourth lines list the edges of graph

Example:

a, b, c, d, e

(a,b); (a,c); (a,a); (e, a);  …. etc….

1, 2, 3, 4, 5

(5, 1); (3,4); (2, 3); (2, 2); … etc …

* Output:
  + One word written to stdout: yes or no depending on whether the two graphs are isomorphic.

Example:

yes

If you choose 13:

* Input:
  + Plaintext file with two lines
  + The first line lists the vertices of the graph
  + The second line lists the edges of the graph

Example:

a, b, c, d, e

(a,b); (a,c); (a,a); (e, a);   …. etc….

* Output:
  + Two lines written to stdout.
  + The word “Circuit” or “Path” or “None”  (newline)
  + The Euler circuit or path, if applicable

Example:

Circuit

a - b - c - e - d - a

Chapter 11:

Implement #8 and #9 in the following way:

You will write a calculator that can compute prefix and postfix notation depending on input. If the input begins with an operator, it’s prefix. If the line begins with a number, it’s postfix.

Note: You shouldn’t store values using ints. If we divide, ints lose decimal values. Do you computations with floats or doubles.

* Input:
  + Plaintext file:
    - One or more lines of computations, properly formatted in pre- or post-fix

Example:

+ 4 - 1 \* 3 5

3 2 + 1 - 8 /

* Output:
  + To stdout:   
    - The answer to each question, one line per question.

Example:

-10

.5

Note: Not all number have good floating point representations. For example, type this into the python interpreter:

>>> 0.1 + 0.2

You’ll see

>>> 0.1+0.2

0.30000000000000004

So that we don’t have to worry about representation in this assignment (that’s not quite the point), if you print to stdout the number 0.30000000000000004, my checker will be smart enough to know that your calculation should be 0.3.  In python you can use the Decimal class that cleans this up a little bit, but again, my checker is looking for

abs(your answer - mine) < 0.001.

Chapter 12: Boolean Algebra   Pick ~~one of problem 2 or~~ problem 4.

Problem 2 is boring. **Only problem 4 will count for credit**. My initial thought is using constructing a tree, so it will be a good practice of chapter 11, as well. (My two reasons for cutting the previously written problem)

~~Problem 2 is the easy one.~~

~~Construct a table of all 256 Boolean functions of degree 3.~~

~~Input: There is no input.~~

~~Output:~~

* ~~A comma-delimited table with nine rows. The first row is a header labeled x, y, z, 1, 2, 3, …., 256  (so 259 columns), and the rows that follow list the values of the possible functions.~~

~~For example, if the problem were to find all 4 functions of degree 1:~~

~~Command line:% ./answerThatQuestion~~

~~x, 1, 2, 3, 4~~

~~1, 1, 1, 0, 0~~

~~0, 1, 0, 1, 0~~

~~That is the complete output of the program. Your program will be a much, much larger output, obviously.~~

Problem 4 is the interesting one.

Given a table of values of a Boolean Function, express the function only in terms of \* and -  (boolean product and complement. No boolean sums!)

Input:

x, y, f

1, 1, 1

1, 0, 0

0, 1, 1

0, 0, 1

Output:

-(-(x\*y)\*x))

(this is one possible answer, there are other representations of the same function. If your’s is equivalent, it will be correct)

\*\*\*\* I hate that output formatting, because it doesn’t quite like like ((xy)x).  Outside of LaTeX, I can’t think of a better way to format the output. I welcome your suggestions if you have a nicer idea!

Chapter 13

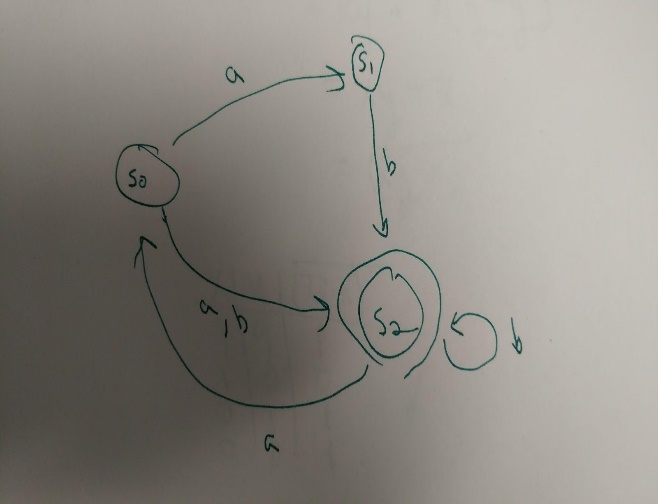
Number 8

Given ~~the state table~~ the set of ordered tuples of a *nondeterministic* finite state automaton and a string, determine whether the string is recognized by the machine.

Thinking about my input:

* I’m choosing to label states s0, s1, s2, ... , s10, s11, …
* ~~The alphabet of the language will be specified in the header row of the table~~
* ~~The rest of the table is~~
* Rather than use a table, I’ll write everything as an ordered tuple (as we do with Turing Machines)
* The format of the tuple is (initial state, next state, input)
* Note that the input piece will define the possible alphabet. It could be letters, it could be numbers

Input:

* A file (passed on the command line!)
* The first line is a string that may or may not be recognized by the language
* The second line is the list of final states (one or more, comma-separated)
* We assume the initial state is s0
* The next n lines (until the end of file) will be ordered tuples which completely determine the FSA.

ababbabab

s2

(s0, s1, a)

(s0, s2, a)

(s0, s2, b)

(s1, s2, b)

(s2, s0, a)

(s2, s2, b)

Output:

* yes or no, depending on whether the string is recognized.
* The answer to the above is yes.