CS350 – Lab 4
Due Date: February 28

Purpose: The big concept of chapter 6 in your text is that there exists a universal Python program (a python program which can execute any other Python program) and a universal Turing Machine (a Turing Machine that can simulate any other Turing Machine). The purpose of this lab is to explore and experiment with these universal machines.

Knowledge: This lab will help you become familiar with the following content knowledge:

- How to simulate a computation using the Universal Python program or Universal Turing Machine.
- How to use the Universal Program/TM machine to alter the effects of a computation.
- How to view decidable and recognizable by using the Universal Program/TM.

Task: Follow the steps in this lab carefully to complete the assignments. You will use files given to you in lab4.zip

Assignment 1:
Here are three different ways of achieving the same computation:

```
>>> containsGAGA('TTGAGATT')
>>> simulateTM(rf('containsGAGA.tm'), 'TTGAGATT')
>>> universal(rf('containsGAGA.py'), 'TTGAGATT')
```

Explain the difference between the above three methods of doing the same computation

Criteria for Success: You have a clear but brief explanation of how each of the three computations above achieve the same result in three different ways and what the fundamental differences are between them.

Given the universal Python program that you just used, we know that there must be a universal Turing machine. This is because any Python program can be converted into a Turing machine so we could just convert this one. Or we could build a universal Turing machine from scratch. Let's look at just such a one.

A universal TM, U, is a TM that simulates other TMs. By providing a description of all transitions for some TM M and the initial contents of a input tape for M, U can simulate M. The file ex-universal is a 3-tape machine which implements a universal TM. The first tape holds a description of M's transitions. The second tape hold M's internal tape. The third tape holds M's internal state.

We first establish some conditions on M. M is a single tape machine with one initial state

q1 and one final state q2 and we will refer to all the other states as q2, q3, ... We will also refer to the TM alphabet as a1 (which will be the blank symbol), a2, a3...

We now will describe a method to encode M's tape symbols, states, and transitions as strings of 0's and 1's. Each state q_i in M is encoded as 1^i (for example q4 is encoded as 1111), and each tape symbol a_j is similarly encoded as 1^j. Finally, we encode the move symbols L, S, and R as 1,11, and 111, respectively. The symbol 0 suffixes each string of 1's. So if the internal tape of M is a3 a5 a2 then this will be encoded as 1110111110110.

The encoding of the transitions requires more explanations. Suppose one of M's transitions is $(q_h, a_i) \rightarrow (q_j, a_k, d_l)$ where d1=L, d2=S, and d3=R. This signifies that if M is in state q_h with the tape head on a_i, then M will overwrite a_i with a_k, move the head in the d_l direction, and move to state q_j. Each transition is encoded as $1^h01^i01^j01^k01^l0$ with all encoded transitions concatenated together to form the machine description for tape1. For example, one would encode $(q1, a3) \rightarrow (q2, a5, S)$ as 101110110111110110 within tape1.

How does ex-universal work? In brief, U executes a continuous loop. First, U performs a linear search in tape1 for a transition that matches the state in tape3 and a symbol at the current head position on tape2. If the current state and symbol do not match the transition, U, proceeds to the next encoded transition in tape1. If the state and symbol match, U changes tape3 to hold the new state and tape2 to overwrite the current symbol with the transition's new symbol, and moves tape2 head in the direction indicated by the transition's direction symbol. Then tape3 is checked to see if it is the halting state (q2) in which case U accepts; otherwise U starts the loop again.

The lengthy encoding makes interesting TMs tedious to encode and input. However, a very simple machine is manageable. The machine ex4.2 is a transducer (it modifies the tape rather than accepting a language) which converts every `a` to `b` and accepts when it reaches the end of the string. Open up this TM and test it.

We can encode the tape alphabet as blank→1, a→11, b→111.
We can encode the transition $(q1, a) \rightarrow (q1, b, R)$ as 101101011101110.
We can encode the second transition as 101011010110.
Therefore the machine description is 10110101110111010101011010110 for tape1.

Suppose we want our initial input to be `aaa`. This we encode as 110110110 for tape2.
Finally, as always tape3 must hold 1 to signify our initial state q1.

Assignment 2:
Run of the universal machine in JFLAP with the input described above. Explain the values on tapes 2 and 3 after the universal machine finishes executing.

Criteria for Success: You have a clear but brief explanation of the meaning of the tape2 and tape3 values after the computation is complete.

The existence of a universal Turing machine means that TMs can now be considered equivalent to general purpose digital computers which can be programmed to do different jobs at different times!

Assignment 3:
Using a universal program, we can alter the effects of other programs in real time. Try the following Python commands:

```
>>> alterGAGAtoTATA(utils.ESS(rf('repeatCAorGA.py'), 'CA'))
>>> alterGAGAtoTATA(utils.ESS(rf('repeatCAorGA.py'), 'GA'))
```

Explain what `alterGAGAtoTATA.py` is doing.

Criteria for Success: You have a clear but brief explanation of how `alterGAGAtoTATA.py` is altering the effect of `repeatCAorGA.py` in real time.

Assignment 4:
Now write your own program which modifies the effect of another program by completing exercise 6.3 on p114.

Criteria for Success: You can test your program by
`>>> repeat(utils.ESS(rf('repeatCAorGA.py'),'GA'))`

Assignment 5:
Take a look at the file `ignoreInput.py` and test it out with:

```
>>> utils.writeFile('progString.txt', rf('containsGAGA.py')
>>> utils.writeFile('inString.txt', 'GGGGTT')
>>> ignoreInput('GAGAGA')
```

Explain what these three commands are doing.

Criteria for Success: You have a clear but brief explanation of how `ignoreInput.py` behaves.

Assignment 6:
The Universal Program can help us understand the difference between decidable (recursive) and recognizable (recursively enumerable) languages. Recall that we already showed that the language `YesOnString` is not decidable. Take a look at the program `recYesOnString.py` and explain why this shows that `YesOnString` is recognizable. Why does this program not also *decide* the language `YesOnString`?

Criteria for Success: You have a clear but brief explanation of how `recYesOnString.py` recognizes but does not decide the language `YesOnString`.

Assignment 7:
Complete exercise 6.9 on p115 in your text.

Criteria for Success: Using the previous assignment as your guide, you provide a way to recognize the language (which happens to be undecidable).

Submit your Python files and written answers in Canvas for grading. You may turn in the written answers on paper if your prefer but the Python files must be submitted in Canvas.