LING 570: Hw3
Due date: 11pm on Oct 19
Total points: 100

Goal: Become familiar with FST.
All the example files mentioned below are under **hw3/examples**/.

Q1 (6 points): Manually create FSTs for the following regular relations and save the FSTs in Carmel format as files "fst1", "fst2", "fst3" under **q1**/.

- fst1 for {(a^{2n}, b^n) | n >=0}
- fst2 for {(a^n, $b^{2n}c$) | n >=0}
- fst3 for{($a^n d*$, $(bc)^n g$) | n>=0}

Q2 (14 points): Use Carmel to build a FST acceptor, **fst_acceptor.sh**.

- The format of the command line is: fst_acceptor.sh fst_file input_file > output_file
- fst_file is an FST in the Carmel format (e.g., "**examples/fst0**", "**examples/wfst1", "examples/wfst2")**
- Each line in the input file is a string (e.g., "**examples/ex**", "**examples/ex2**")
- Each line in the output_file has the format "x => y prob" (e.g., "**examples/ex.fst0**"), where
  - x is the string from the input file.
  - y is the output string if x is accepted by the FST, or *none* if x is not accepted by the FST.
  - prob is the probability of the path whose yield is x.
  - The probability of a path is the product of the probabilities of the edges in the path.
  - If there are multiple paths for an input string x, y is the output string of the path with the highest probability (for paths with the same probabilities, Carmel breaks the tie somehow)

- Run your fst_acceptor.sh with the FSTs in Q1 and hw3/examples/ex as input file, save the output files in ex.fst[1-3], respectively, under **q2/.**

```
fst_acceptor.sh   q1/fst1    hw3/examples/ex >  q2/ex.fst1
fst_acceptor.sh   q1/fst2    hw3/examples/ex >  q2/ex.fst2
fst_acceptor.sh   q1/fst3    hw3/examples/ex >  q2/ex.fst3
```

- Run the following commands and save the output files under **q2/.**

```
fst_acceptor.sh   hw3/examples/wfst1  hw3/examples/ex2 >  q2/ex2.wfst1
fst_acceptor.sh   hw3/examples/wfst2  hw3/examples/ex2 >  q2/ex2.wfst2
```

Q3 (55 points): Build **fst_acceptor2.sh** WITHOUT using Carmel, which has the same command line format and functionality as fst_acceptor.sh. The only differences are:

- fst_acceptor2.sh CANNOT use Carmel
- For the sake of simplicity, fst_acceptor2.sh can assume that the input FST does NOT contain epsilon arcs (arcs whose input symbols are empty strings).

a) Note that the input FST might be nondeterministic. Unlike FSA, not every non-deterministic FST can be converted to a deterministic one. So your code needs to use Viterbi Algorithm as discussed in class.

b) Since the code takes an FST not an FSA, you need to extend that algorithm a little bit to find out the output sequence for the best path for the input. In your note file, briefly explain:

   - what data structure you use to store the input FST
   - what changes you need to make to the Viterbi algorithm to handle FST

c) Run the following commands and save the output files under **q3/.**

```
fst_acceptor2.sh   hw3/examples/wfst1  hw3/examples/ex2 >  q3/ex2.wfst1

fst_acceptor2.sh   hw3/examples/wfst2  hw3/examples/ex2 >  q3/ex2.wfst2
```

The submission should include:

- The readme.(pdf | txt) file that includes your answer to Q3(b).
- Hw.tar.gz that includes all the files specified in submit-file-list