Assignment 3

Due Friday, March 17th, 2017

This assignment will be submitted electronically via OWL. See below for submission instructions.

1. [10 marks] Draw the Turing Machine Diagram

Create the state diagram of a Turing machine deciding the following language:

$$L = \{a^i b^j c^k \mid k = i * j, \text{ and } i, j, k \geq 1\}$$

2. [20 marks] Write a Turing Machine Interpreter

For this part of the assignment, you will create a Turing machine interpreter in C/C++, Java, or Python (your choice).

Input/Output Format

The program will read from standard input and write to standard output. Standard input will consist of a filename. The input file should contain the following:

(a) A specification of the behavior of the Turing machine.

(b) A number of test input strings to be run by the Turing machine.

For each test string, the output will write:

(a) Whether the Turing machine accepts or rejects,

(b) The contents of the tape after it halts.

Recall a Turing machine is specified by a set of transitions: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$. Consider the following example transition:

$$q_0, 0 \rightarrow q_1, 1, R$$

The transition specifies that if the machine is currently in state q_0 and the head is currently reading a 0, then do 3 things: update the machine to state q_1, write a 1 to the tape, then move the head one step to the right.

Transitions. The format of the transitions to be input into your Turing machine emulator will be as follows:

t *current state* *input symbol* *next state* *write symbol* *head movement direction*

So for example, the transition above would be written as follows:

```
t 0 0 1 1 R
```

Final/accept states. The start state is numbered `0`. Final (accept) states are indicated using a line formatted as follows:

f *final state 1* *final state 2* ... *final state n*

So for example, if states 4 and 7 of were accept states of this particular Turing machine, you would write:

```
f 4 7
```

Input tape. Input to the Turing machine (i.e., the initial tape contents) is a line beginning with 'i' followed by a string of symbols. You can assume that an infinite sequence of blank symbols (i.e., ␣) extend in both directions. For example, the following input:

```
i 10101
```

would be used to specify the following tape:

...	␣	␣	1	0	1	0	1	␣	␣	...

Output. Once your computation is complete (an accept or reject state was encountered), output the final contents and the accept/reject status. For example:

```
10101
ACCEPT
```

Some Q&A

Question: Do we need to explicitly specify an input and tape alphabet? Do we need to specify reject states?
Answer: No and no. Your Turing machine should be able to compute its output based on the transitions and final state inputs alone. You can assume that if the machine ever encounters a state/input pair that is *not* in your list of transitions, that the machine will halt and output *REJECT*.

Question: You said we have to output if the machine accepts or rejects, but what if you feed it an input that causes it to go into an infinite loop?
Answer: You can assume we will feed your machine inputs that would always cause a correctly implemented machine to halt.

Question: How should we represent the blank character in a transition expression?
Answer: For the purposes of this assignment, we will sue the underscore symbol '_'.

3. [15 marks] Create a Turing Machine Program

Using the notation from part 1, create an input file that specifies a Turing machine that accepts a binary encoded integer n and outputs a binary encoded integer $n - 1$. Note: this input file will be tested on a program designed to the specifications outlined in Part 1; it will not use the program you submit in Part 1.

The decrement operation will be assumed to be limited to inputs of 8 bits in length. Input will include all 8 bits of a binary number n (with the most significant bit on the left), and the output will be $n - 1$, over-written on to the input. Here are some example inputs and outputs:

Input:

```
i 00000001
i 01100101
i 11010110
```

Output:

```
00000000
01100100
11010101
```

Specify all of the transitions that are required to implement this functionality using the same format as in Part 1:

t *current state* *input symbol* *next state* *write symbol* *head movement direction*
t . . .
⋮

Put these lines into a file and submit them.

4. Submission Instructions

Submission for assignment 3 will be done through the OWL (email submissions will not be accepted). Place all files into a zip file called `assigment3.zip`.

What files should I include?

Include all of your code. Additionally, include any makefiles/build commands/project files required to build your project on an external machine. Executable files are **not** permitted (jar files are acceptable if using java). The test platform will be Unix-based, but solutions in Windows formats are acceptable. It might be a good idea to package your files and test them on an external machine to ensure that everything works fine and isn't depending on anything local.

How will the program be tested?

Your program will be tested as a standalone program using standard input and output. In a Unix bash shell, the testing will be as follows:

```
$ YourProgram < input.txt > output.txt
```

Here, '<' redirects the standard input to come from the specified file (input.txt) and '>' redirects the output to another file.

Is the Turing tape infinite in length?

Technically, yes; in our case we will assume that 'Z' symbols (always capitalized) follow and precede our input symbols like so:

$$...ZZZabbaabbbZZZ...$$

Practically, you will only ever have to examine the two Z's that bound the input - the one to the left and the one to the right of the string: ZabbaabbbZ. If you are using a different approach (not recommended) make it clear to me in your email message.

Anything else?

Yes, here are some tips:

- States will always be integers (0, 1, 2 . . .)
- Input symbols will always be within the ASCII character set, specifically a-z and 0-9
- Input lines in test files will always be ordered with transitions first (t) followed by accepting states (f) and finally input strings (i)
- If there is an input symbol with no transition on the current state, the string is rejected!
- Part marks will be given even for incomplete code! Hand in what you have

Attached you will find sample input and output files. The input specifies the language aba and a number of test strings. Note that the tape also turns the first 'a' into an 'x' (assuming the input begins with an 'a').

If you have any questions, email Rhys: `rcarlton(at)uwo.ca`