Assignment 7

Note, for the first two questions the Turing machine must be provided as a list of transitions using the following notation:

STATE,SYMBOL -> STATE,SYMBOL,DIRECTION

use underscore (_) for a blank. Use l and r for direction (left / right). You may also use s for stay moves. An example is the following:

```
S,_ -> FE,_,r
FE,_ -> ha,_,s
FE,a -> FE,_,r
FE,b -> FE,_,r
```

which erases the tape. You may also use * for a wild card: FE,* -> FE,_,r will replace whatever it sees with a blank, and FE,* -> FE,*,r will write the same symbol as it sees (i.e., doesn't change the contents of the current cell). The rules are processed in order from top to bottom, so the first rule in the list that matches the state-symbol pair (e.g., wildcard) will be used even if a later rule is more refined for the symbol.

You may use the online tool at `http://pages.cpsc.ucalgary.ca/~joel.reardon/313/tools/tm/tm.html` to both test your design and ensure your syntax is correct. You may also provide a written description or a state-machine style diagram if that helps you in the design process. Use the \begin{verbatim} and \end{verbatim} enviroment in LaTeX if you submit the description in latex.

1. Provide a Turing machine that does the following operation: find the middle letter of a word. It should take as input a word, find the middle symbol and halt and accept. It should not change the word. It should halt and reject words that have an even number of letters. It should halt and accept words that have an odd number of letters. If the input word is *aabaa* then it should halt and accept with the tape head on the symbol *b*. You may use left/right/stay moves in your machine. Points will be deducted for having more than 10 states.

   The Turing machine finds the middle letter of a word is:

   ```
   S,_ -> A,_,r
   A,a -> B,x,r
   A,b -> B,y,r
   A,x -> hr,x,s
   A,y -> hr,y,s
   B,a -> B,a,r
   B,b -> B,b,r
   B,* -> C,*,l
   C,a -> D,x,l
   C,b -> D,y,l
   ```

```
C,x -> E,A,l
C,y -> E,B,l
D,a -> D,a,l
D,b -> D,b,l
D,x -> A,x,r
D,y -> A,y,r
E,x -> E,a,l
E,y -> E,b,l
E,_ -> F,_,r
F,x -> F,a,r
F,y -> F,b,r
F,a -> F,a,r
F,b -> F,b,r
F,A -> F,A,r
F,B -> F,B,r
F,_ -> G,_,l
G,A -> ha,a,s
G,B -> ha,b,s
G,* -> G,*,l
```

2. Provide a Turing machine that does the following operation: replace the second half of a word with its upper-case version. For example, if the input is *aaab* then it should halt and accept with the tape containing *aaAB*. If the word has odd length it should halt and reject.

   The Turing machine for this operation is:

```
S,_ -> A,_,r
A,x -> A,x,r
A,y -> A,y,r
A,a -> B,x,r
A,b -> B,y,r
A,A -> E,A,l
A,B -> E,B,l
E,a -> E,a,l
E,b -> E,b,l
E,x -> E,a,l
E,y -> E,b,l
E,_ -> ha,_,s
B,a -> B,a,r
B,b -> B,b,r
B,* -> C,*,l
C,a -> D,A,l
C,b -> D,B,l
C,x -> hr,A,s
C,y -> hr,B,s
```

```
D,a -> D,a,l
D,b -> D,b,l
D,x -> A,x,r
D,y -> A,y,r
```

3. Provide a Turing machine that does the following operation: whatever you want. These are universal computers so anything is possible. Be creative. Describe what your machine does as through it were a function, as in, what is the input and what is the output. Is it a function that mutates the tape, or is it a decision problem that answers a question?

   The following Turing machine multiply 2 numbers. The machine takes 2 unary numbers as inputs (Ex: 0010 which means 2 and 1) and gives out output is the product of 2 numbers (Ex: 00). The result will keep the first number, change all 0s of the second number to x, followed by a blank and product (Ex: 001$x_0$0).

   When the machine reaches 0 of the second number, change 0 to x and copy all 0 from the first number to the right of inputs after the blank. The tape repeats copying the first number until all 0s in second number become x.

```
S,_ -> A,_,r
A,0 -> A,0,r
A,1 -> A,1,r
A,_ -> B,1,l
B,0 -> B,0,l
B,1 -> C,1,r
C,x -> C,x,r
C,0 -> D,x,l
C,1 -> ha,_,s
D,x -> D,x,l
D,1 -> E,1,l
E,y -> E,y,l
E,0 -> F,y,r
E,_ -> K,_,r
F,y -> F,y,r
F,1 -> G,1,r
G,0 -> G,0,r
G,x -> G,x,r
G,1 -> H,1,r
H,0 -> H,0,r
H,_ -> I,0,l
I,0 -> I,0,l
I,1 -> J,1,l
```

```
J,0 -> J,0,l
J,x -> J,x,l
J,1 -> E,1,l
K,y -> K,0,r
K,1 -> C,1,r
```