Question 1: A Post-Turing Program for Caesar’s Code

Caesar’s code is probably the simplest way of encoding a message, and it can also be cracked very easily. Let us consider its easiest form for an alphabet \( A = \{a, b, c\} \). For a given input string, we replace each letter ‘a’ with ‘b’, each letter ‘b’ with ‘c’, and each letter ‘c’ with ‘a’. The resulting string is the encoded message.

Write a Post-Turing program that computes this encoding for a given input string strictly.

```
RIGHT
IF B GOTO E // If input is empty, then output is empty, too.
[D1] IF a GOTO A
    IF b GOTO B
    IF c GOTO C
[D2] LEFT // When done, go to left of input for strict computation
    IF B GOTO E
    GOTO D2
[A] PRINT b // Write the modified letter, go right, and continue
    GOTO D3
[B] PRINT c
    GOTO D3
[C] PRINT a
[D3] RIGHT
    GOTO D1
```
Question 2: From Quadruples to Quintuples

Consider the following Turing machine on the alphabet $A = \{a, b\}$:

\[
\begin{align*}
q_1 & \quad a & \quad R & \quad q_2 \\
q_1 & \quad b & \quad R & \quad q_2 \\
q_1 & \quad B & \quad R & \quad q_2 \\
q_2 & \quad a & \quad b & \quad q_1 \\
q_2 & \quad b & \quad a & \quad q_1 \\
q_2 & \quad B & \quad B & \quad q_2 \\
q_2 & \quad B & \quad B & \quad q_2 \\
q_2 & \quad B & \quad B & \quad q_2
\end{align*}
\]

(a) Describe in words the function that this machine computes. Is it a strict computation?

This machine replaces all ‘a’s with ‘b’s and all ‘b’s with ‘a’s. The tape head ends up on the right of the output string rather than on its left, so it is not a strict computation.

(b) Translate this machine into an equivalent quintuple Turing machine using exactly the scheme that we discussed in class. Write down all resulting quintuples (no state transition diagram necessary).

According to the scheme, the quintuple Turing machine has four states, even though one of them ($q_4$) is never used:

\[
\begin{align*}
q_1 & \quad a & \quad a & \quad R & \quad q_2 \\
q_1 & \quad b & \quad b & \quad R & \quad q_2 \\
q_1 & \quad B & \quad B & \quad R & \quad q_2 \\
q_2 & \quad a & \quad b & \quad R & \quad q_3 \\
q_2 & \quad b & \quad a & \quad R & \quad q_3 \\
q_3 & \quad a & \quad a & \quad L & \quad q_1 \\
q_3 & \quad b & \quad b & \quad L & \quad q_1 \\
q_3 & \quad B & \quad B & \quad L & \quad q_1 \\
q_4 & \quad a & \quad a & \quad L & \quad q_2 \\
q_4 & \quad b & \quad b & \quad L & \quad q_2 \\
q_4 & \quad B & \quad B & \quad L & \quad q_2
\end{align*}
\]
Question 3: The Turing Machine Competition!

Build a Turing machine on the alphabet $A = \{a, b\}$ that computes a function $f(x)$ strictly. $f(x)$ outputs only the symbol “a” if $x$ is a palindrome and outputs only the symbol “b” otherwise. A palindrome is a symmetrical word, i.e., if we reverse the order of its letters, it is still the same word. For example

\[
\begin{align*}
  f(bba) &= b \\
  f(baab) &= a \\
  f(abab) &= b \\
  f(aaa) &= a \\
  f(babab) &= a \\
  f(0) &= a
\end{align*}
\]

**Hint:** Your machine could start by checking the first symbol of the string and, depending on the result (‘a’ vs. ‘b’), take one of two paths. In each path, it first deletes the current symbol, moves all the way to the rightmost symbol of the string and checks whether it is the same symbol as the original leftmost one. If so, the program deletes the rightmost symbol, goes back to the (new) leftmost symbol, and repeats the whole thing. Otherwise, it removes all symbols from the tape, outputs “b”, and terminates. If no more symbols are on the tape without having found any mismatch, the program outputs “a” and terminates.

Write down the Turing machine in quadruple notation and as a state transition diagram. Also give the sequence of configurations during the computation of $f(aba)$.

Whoever builds the Turing machine with the fewest internal states that correctly computes $f$ will get some bonus points!

You can test your machines using the Haskell code that I will upload.

The machine on the next page implements exactly the algorithm described in the hint above. Starting on the left, the upper path is chosen if he leftmost symbol is an ‘a’ and the lower path for a ‘b’. If a match with the rightmost symbol is found, we end up in state $q_9$, and states $q_{10}$ and $q_1$ are used to move back to the leftmost symbol. If a mismatch is found, we get to state $q_11$, and together with state $q_{12}$ it erases the entire tape content and places a ‘b’ on the tape. If no mismatch is ever found, an ‘a’ is written on the tape. The algorithm always terminates in state $q_{13}$ which moves the tape head one position to the left of the output symbol as required for a strict computation.
q1  a  L  q1
q1  b  L  q1
q1  B  R  q2
q2  a  B  q3
q2  b  B  q6
q2  B  a  q13
q3  B  R  q4
q4  a  R  q4
q4  b  R  q4
q4  B  L  q5
q5  a  B  q9
q5  b  B  q11
q5  B  a  q13
q6  B  R  q7
q7  a  R  q7
q7  b  R  q7
q7  B  L  q8
q8  a  B  q11
q8  b  B  q9
q8  B  a  q13
q9  B  L  q10
q10  a  L  q1
q10  b  L  q1
q10  B  a  q13
q11  B  L  q12
q12  a  B  q11
q12  b  B  q11
q12  B  b  q13
q13  a  L  q13
q13  b  L  q13