CPSC 351 — Tutorial Exercise #6
Introduction to Turing Machines

About This Exercise

The goal of this exercise is to help you to check that you understand how Turing machines process strings, and to give you practice designing Turing machines and proving their correctness.

Problems To Be Solved

Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. A string

$$\omega = \alpha_1 \alpha_2 \ldots \alpha_n$$

(with length $n \geq 0$, so that $\alpha_1, \alpha_2, \ldots, \alpha_n \in \Sigma$) is a ***palindrome*** if $\omega = \alpha_n \alpha_{n-1} \ldots \alpha_1$ as well — that is, you get the same string by reading the symbols from right to left instead of from left to right.[1] Let

$$L = \{\omega \in \Sigma^\star \mid \omega \text{ is a palindrome}\}.$$

1. Consider the "`isPal`" algorithm that is shown in Figure 1 on page 2. Prove that an execution of this algorithm on an input string $\omega \in \Sigma^\star$ ends after a finite number of steps. Furthermore, the algorithm returns `true` if $\omega \in L$ and it returns `false` if $\omega \notin L$.

2. Use this algorithm to produce an ***implementation-level description*** of a Turing machine M, with input alphabet Σ, which satisfies the following properties — where q_0 is the start state for M:

   i. The Turing machine halts, after a finite number of steps, when the execution continues from a configuration

   $$\sqcup^h \, q_0 \, \omega \tag{1}$$

   — for every non-negative integer h and for every string $\omega \in \Sigma^\star$.

   ii. If $\omega \in L$ and the Turing machine continues an execution from the configuration shown at line (1), for any non-negative integer h, then the Turing machine reaches an ***accepting*** configuration after finitely many more steps.

   iii. If $\omega \notin L$ and the Turing machine continues an execution from the configuration shown at line (1), for any non-negative integer h, then the Turing machine reaches a ***rejecting*** configuration after finitely many more steps.

[1] Another way to say this is that the string is its own ***reversal***.

```
boolean isPal (ω : Σ*) {
 1.  if (ω == λ) {
 2.    return true
 3.  } else if (ω begins with "a") {
 4.    if (|ω| == 1) {
 5.      return true
 6.    } else if (ω == a · μ · a for some string μ ∈ Σ*) {
 7.      return isPal(μ)
       } else {
 8.      return false
       }
     } else {
 9.    if (ω == 1) {
10.      return true
11.    } else if (ω == b · μ · μ for some string μ ∈ Σ*) {
12.      return isPal(μ)
       } else {
13,      return false
       }
     }
}
```

Figure 1: An Algorithm to Decide Membership in the Language L

Describe how your implementation-based description has been produced using the "`isPal`" algorithm that is given above.

3. Give a ***formal description*** for a Turing machine M, with input alphabet M, that is based on your implementation-level description and that decides the language L — describing how your formal description is based on your implementation-level description.

4. Sketch a proof that your Turing machine decides L — by giving a sequence of lemmas (concerning various computations of L), describing how they can be proved, and then giving a proof, that M decides L, using them.