1. **Part (a): 5 marks. Part (b): 5 marks.**

   Part (a) of this problem was done perfectly by all students. Some even mentioned the AKS primality test; if you haven’t heard of that, it’s quite an interesting algorithm!

   Part (b) wasn’t done as well. Only four students gave the correct answer; the problem is decidable! Below, I reproduce the answer provided by Prof. Salomaa:

   Let \( y \) be the maximum number of atoms in the universe at any time (possibly \( y = \infty \)).

   The algorithm simply checks whether or not \( x \leq y \). Note: We have no way of knowing what \( y \) is. However, the number \( y \) exists (possibly \( y \) is infinite), and the algorithm using this \( y \) decides the problem.

   Admittedly, this question fooled me as well, so I wasn’t overly harsh when marking it. If you answered “undecidable”, I deducted half-marks (2.5 out of 5).

2. **Part (a): 2.5 marks. Part (b): 2.5 marks. Part (c): 2.5 marks. Part (d): 2.5 marks.**

   All parts of this problem were done very well.

   I noticed two recurring (minor) mistakes. First, be mindful of blank cells on the tape, and try to denote them using a symbol like \( \cdot \). Second, a “sequence of configurations” is different from a “sequence of states”. When we ask for configurations, we’re interested in seeing how the tape looks at each step of the computation; the current state is only one part of the configuration.

3. **Sequence of configurations: 5 marks. Language recognized: 5 marks.**

   This problem was done rather well. The most common mistakes were giving an incorrect language as a solution or not noticing that the empty string was recognized by the given Turing machine. Another less common mistake was confusing the contents of the work tape for the strings in the recognized language. Some students forgot to write the sequence of configurations for the given string.

   As a side note, I saw many students write languages in the form of \( \{(ab)^n \mid n \geq 0\} \). This is correct, but it can be expressed in an easier way using the Kleene star by writing \( \{(ab)^*\} \).

4. **Machine construction: 5 marks. Sequence of configurations: 5 marks.**

   This problem was also done rather well. The most common mistakes were forgetting to account for palindromes of odd length (which just required two additional transitions) and forgetting to write the sequence of configurations for the given string. I didn’t deduct marks for forgetting the “odd-length palindrome” transitions, but in the future, remember that it’s important to account for all cases when constructing a machine for a given language.
5. Whole question: 10 marks.

I roughly broke up the marking of this question into two parts: identifying a way of finding the “middle” of a string, and verifying whether or not a string is a square after having found the “middle”.

Many students made the assumption that their Turing machine can count the number of tape cells occupied by the string. In order to do this, you would need to implement some sort of counter on the work tape; a Turing machine cannot count tape cells on its own. I didn’t deduct marks if you used a counter, but it’s a detail to keep in mind for the future.

6. Whole question: 10 marks.

Students generally took one of two approaches for this problem: simulating a left move either by shifting the contents of the entire tape or by using markers to determine the tape head’s position within the tape. If you gave a correct answer with sufficient justification, you received full marks.

Questions/comments? Feel free to stop by my office hours or send me an email at tsmith [at] cs [dot] queensu [dot] ca.