COSC 341 – Assignment 2

Due: Wednesday, May 7, 11:59 p.m.

Instructions: Please submit a PDF file of your solutions, along with a text file of your Turing machine code via Blackboard. Point values for each question are indicated in parentheses thus: (**0**).

Even if a question only seems to ask for an answer e.g., "How many …", an explanation of the reason the answer is correct is also required.

Despite the fact that there appear to be 11 points, the maximum number of points available for this assignment is 10, corresponding to the 10% weight that it has in your final grade. In other words, if you tackle all questions, and only lose one mark, then you will get a perfect score of 10.

1. (**2**) Show, using the pumping lemma for regular languages or the Myhill-Nerode theorem, that (a):
$$L = \{a^n b^m \mid 0 \leqslant n,\, 0 \leqslant m \leqslant 2n\}$$
is not regular. (b) Describe a pushdown automaton, M, such that $L = L(M)$, and (c) give a context-free grammar for L.

2. (**2**) Show, using the pumping lemma for context free languages, that
$$L = \{a^n b^m \mid 0 \leqslant n,\, m = n!\}$$
is not context free.

3. (**3**) Design a (standard) Turing machine to accept the language over the alphabet $\{a, b, \#\}$ consisting of all words of the form $w\#v$ for where $w \in \{a, b\}^*$ is arbitrary and v is the concatenation of 0 or more copies of w. So it should accept all of:
$$a\#,\ a\#aaaa,\ ab\#,\ ab\#ab,\ ab\#abababababababababab,$$
and none of:
$$a\#b,\ a\#aaba,\ ab\#a,\ ab\#aba,\ ab\#ababababbbabababab.$$
As well as a high-level description of how the machine operates, please submit a text file suitable for use in Anthony Morphett's Turing machine simulator.

4. (**1**) Give an explicit reduction from Halt to show that there is no algorithm to decide whether a Turing machine will halt exactly on inputs of the form $1111\cdots1$.

5. (**2**) Consider the following decision problem:

   Halt in Bounded Space
   Instance: A Turing machine M, and a word w.
   Problem: Does M halt on input w without ever moving the read/write head beyond the final character of w?

   Show that there is an algorithm for solving Halt in Bounded Space[1].

6. (**1**) Show that the problem of determining whether a word $w \in \{0, 1\}^*$, thought of as representing a non-negative integer written in binary, is equal to twice a perfect square belongs to **P** (you can argue at the level of "real" algorithms or programming languages rather than Turing machines)

[1]**Hint**: Under what circumstances can we be certain that we have entered a loop?