CSCI 3130: Formal Languages and Automata Theory — Homework 5
The Chinese University of Hong Kong, Fall 2011 — due Tuesday 29 November

Each of the problems is worth 10 points. Write your name, student ID, and your TA's name on the solution sheet

Please write your solutions clearly and concisely. If you do not explain your answer you will be given no credit. You are encouraged to collaborate on the homework, but you must write your own solutions and list your collaborators on your solution sheet. Copying someone else's solution or solutions found in external references will be considered plagiarism and may result in failing the whole course.

Please turn in the solutions by 11.59pm on Tuesday 29 November. The homework should be dropped into the box labeled CSCI 3130 on the 9th floor of SHB. Late homeworks will not be accepted.

Problem 1

Show that the following languages are decidable.

(a) $L_1 = \{\langle R \rangle \colon R$ generates at least one string w that has `111` as a substring$\}$.
Here, R is a regular expression over alphabet $\{\mathtt{0}, \mathtt{1}\}$.

(b) $L_2 = \{\langle G, k \rangle \colon G$ generates some string $\mathtt{1}^n$ where $n \leq k\}$.
Here, G is a context free grammar over alphabet $\{\mathtt{1}\}$.

(c) **(Extra credit)** $L_3 = \{\langle G, k \rangle \colon G$ generates some string $\mathtt{1}^n$ where $n \geq k\}$.
Here, G is a context free grammar over alphabet $\{\mathtt{1}\}$.

Problem 2

For each of these languages, say whether it is decidable. Justify your answer. Here, M, M_1, and M_2 are all Turing Machines.

(a) $L_1 = \{\langle M \rangle \colon M$ accepts the input `0` within 999 transitions$\}$.

(b) $L_2 = \{\langle M \rangle \colon M$ does not accept the input `0`$\}$.

(c) $L_3 = \{\langle M \rangle \colon M$ accepts a finite number of inputs$\}$.

(d) **(Extra credit)** $L_4 = \{\langle M_1, M_2 \rangle \colon M_1$ rejects input $\langle M_2 \rangle$ or M_2 accepts input $\langle M_1 \rangle$ (or both).$\}$

Problem 3

For each of the following variants of the Post Correspondence Problem (PCP), say if it is decidable or not. Justify your answer by describing a decider, or by reducing from (standard) PCP.

(a) $PCP_\star = \{\langle P \rangle \colon P$ is in PCP and every tile in P is a $\star$-tile.$\}$.
A $\star$-tile is a tile where both the top and bottom strings begin with $\star$, for example

$\begin{array}{|c|}\hline \star\mathtt{ab}\star\mathtt{c} \\ \star \\ \hline\end{array}$ is a $\star$-tile, but $\begin{array}{|c|}\hline \mathtt{a}\star \\ \star \\ \hline\end{array}$ is not.

(b) $PCP_1 = \{\langle P \rangle \colon P$ is in PCP and every tile in P is a one-symbol tile.$\}$
A one-symbol tile is a tile where the top and bottom strings have length at most one, so

$\begin{array}{|c|}\hline \mathtt{a} \\ \varepsilon \\ \hline\end{array}$ $\begin{array}{|c|}\hline \varepsilon \\ \mathtt{b} \\ \hline\end{array}$ $\begin{array}{|c|}\hline \mathtt{a} \\ \mathtt{b} \\ \hline\end{array}$ are all one-symbol tiles.

Problem 4

You just got hired by **Doodle**, the hottest software company of the 2010s. Your new boss has several project proposals for you. However, you suspect that some of her proposals may be a bit unrealistic. But you won't turn down a proposal just because you don't like it, or else you might get fired pretty soon. You have to give a reason why you think it is not going to work.

For each of these software projects, say if you think the project is feasible or not. If you think it is feasible, say how you would approach it. If you think it is infeasible, explain why.

(a) Doodle just came up with a new programming language called **Plum**. However most of their old programs, and there are many of them, are written in Java. Write an application which converts all their Java programs into Plum.

(b) Doodle solicits applications from developers that it then sells in its Doodle Store. But some of these developers are very clumsy and their applications tend to crash. It would be nice to have a tool that detects these crashing applications so Doodle can take them out of their store. Write a program called **crash detector**, which looks at the code of an application (written in Plum) and figures out if the application will ever crash.

(c) Some programs take a very long time to run. It would be nice to know roughly how long a program is going to run for ahead of time, so if it takes a long time you can go out and get lunch. Write an application called **timer**, that looks at a computer program and gives an estimate of its running time.