1. Assume that ZF set theory (the standard axiom system for mathematics) is sound. Let $M$ be a Turing machine that, on input $x$ in $\{0, 1\}^n$, first checks whether $x$ (or any contiguous substring of $x$) encodes a ZF proof that ZF is inconsistent, and then runs for $3^n$ steps if so, and for $2^n$ steps otherwise.

(a) **What is $M$’s asymptotic running time?**

Since we are assuming ZF set theory is sound, $x$ will not encode a proof that ZF is inconsistent. Thus, $M$ will run for $O(2^n)$ steps.

(b) **Is $M$’s asymptotic running time provable in ZF? Why or why not?**

If one could prove in ZF $M$’s asymptotic time, one prove ZF’s consistency in ZF. Since we assumed ZF is sound, by G"odel’s Theorem, we cannot prove its consistency.

2. Let a *tile* be a 1-by-1 square, each of whose four edges has a color (chosen from some finite set of colors). Two tiles “snap together” along an edge, if and only if that edge has the same color on both of the tiles. The tiles are all aligned the same way, and cannot be rotated. Then given a finite set of tiles $C$, say that $C$ “fills the $n$-square,” if it’s possible to fill a square of size $n$-by-$n$ with $n^2$ tiles from $C$ (where each tile in $C$ can be duplicated an unlimited number of times), so that every two neighboring tiles snap together.

(a) **Call a set of tiles $C$ good if $C$ fills the $n$-square for every positive integer $n$. Let $TILE$ be the problem of deciding whether $C$ is good, given a description of $C$ as input. Show that $TILE$ is Turing-reducible to $HALT$.**

A Turing machine $M^{HALT}$ that has the $HALT$ oracle can solve $TILE$ by constructing another Turing Machine $T$ that works as follows:

for $n$ from 1 to $\infty$:

i. Try all combinations to see if $C$ can fill the $n$-square.
ii. Halt if it cannot.

$M^{HALT}$ can then check if $T$ halts, and if so, reject (since that means that $C$ is not good) and accept otherwise.

(b) **Show that a set of tiles $C$ is good, if and only if it’s possible to fill the entire infinite plane using tiles from $C$.**

Consider the “0 × 0” square as the root of a tree. From it, for each tile in $C$ there is a branch, so that the next level represents the $1 \times 1$ level. From each of those there are more branches to the $2 \times 2$ level. Assuming that the tree can continue in this way, by König’s Lemma, we know that for each $n$ there will always be at least one branch that is the ancestor of the $(n + 1) \times (n + 1)$ level, meaning there will be an infinite path. Note that for each $n$, there may be one or more $n \times n$ branches; what we know is that at least one of them will be the one on this infinite path.
3. Let $L$ be a subset of $0,1^n$. Since $L$ is finite, it is clear that $L$ is regular. Nevertheless, by using a variant of Shannon’s counting argument, show that it is possible to choose an $L$ so that any DFA that recognizes $L$ must have $\Omega\left(\frac{2^n}{n}\right)$ states.

There are $2^n$ $n$-bit strings. For each of those strings, a given language will either include it or not include it — thus, there are $2^{2^n}$ total possibilities for $L$.

Consider a DFA with $T$ states. Each of these states can connect to any other state, which means that the total number of $T$-state DFA’s is approximately $T^T$. Since a DFA decides one particular language, by the pigeonhole principle, it must be the case that there are at least as many DFA’s as languages; in other words,

\[
2^{2^n} \leq T^T \\
\log_2 2^{2^n} \leq \log_2 T^T \\
2^n \leq T \log_2 T
\]

Plugging in $\frac{2^n}{n}$ for $T$ and performing the same approximations as used in Shannon’s argument, one can see that $T$ must indeed be $\Omega\left(\frac{2^n}{n}\right)$ for the inequality to hold.

4. Give a context-free grammar that generates the complement of the language $L = \{a^m b^n \mid m, n > 0\}$.

Note that $L$ is a regular language, meaning its complement is regular, too. It is not hard to come up with a DFA for $L$, after which we can simply switch all accepting and non-accepting states to come up with a DFA for $\overline{L}$. Then, we can make a regular expression for $\overline{L}$ and from that come up with the following grammar:

\[
S \rightarrow \varepsilon \mid A \mid bC \mid ABaC \\
A \rightarrow aA \mid \varepsilon \\
B \rightarrow bB \mid \varepsilon \\
C \rightarrow aC \mid bC \mid \varepsilon
\]