1. We consider two different languages that in some sense represent strings representing correct additive equalities. You can assume that the input is on a one-way read-only tape. Show that one of these problems can be solved by a finite state machine. Show from first principles (so no using results like the Pumping Lemma) that one of these problems can not be solved by a finite state machine. To show impossibility, you should assume that there is a finite automata $M$ with $k$ states that accepts this language. Then consider arbitrarily large strings of some particular type, and argue $M$ has to mess up on some string of length greater than $k$.

(a) The language consists of strings over the symbols: 0, 1, + and =. The languages consists of strings of the form $x + y = z$, where natural numbers $x$, $y$ and $z$ encoded in binary in the standard way, and where $x$ plus $y$ is indeed equal to $z$. So for example the string $10100 + 111 = 11011$ of 11 symbols is in the language.

(b) The language is a string of the following 8 symbols (so each line is a different symbol)

- [000]
- [001]
- [010]
- [011]
- [100]
- [101]
- [110]
- [111]

Note the pattern here is that the third bit is equal to the sum of the first two bits modulo 2. So to reuse our example from the previous subproblem, the 5 symbol input

- [101][001][110][011][011]

is in the language because $10100 + 00111 = 11011$. Note that in this 5 symbol encoding of this summation equation, the $k^{th}$ symbol is determined by the $k^{th}$ bit in each of the three numbers in the equation.

So this shows that whether one whether a finite state machine can add, depends on the definition of notion of addition that you use.

Due Monday January 14 (You need not use LaTeX, your solutions may be handwritten.)

2. Problem 1.1 from the text. That is, construct a complete formal description of a Turing machine (using the formalization given in section 1.2) that adds two binary numbers (you need not do multiplication). Assume that input is of the form NUMBER1#NUMBER2, e.g. 1011#110. You should write your output on a second tape.

Due Monday January 14 (You need not use LaTeX, your solutions may be handwritten.)
3. (a) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ is the empty language.

(b) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ is the language of every string over the input alphabet.

(c) (warm up) Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ includes the string 11110.

(d) Let $P$ be some property of languages. Further assume there is a Turing machine $M_1$ that accepts a language $L_1$ that has property $P$, and a Turing machine $M_2$ that accepts a language $L_2$ that does not have has property $P$. Show by reduction from the Halting Problem that there is no Turing machine that takes as input a Turing machine $M$, and determines whether the language $L(M)$ accepted by $M$ satisfies property $P$.

(e) Explain why the first three subproblems are consequences of the fourth subproblem.

Due Wednesday January 16. Note that some students may find the fourth subproblem somewhat tricky. Do the other subproblems first, and then make a good faith effort at the fourth subproblem. At the very least, at least understand the statement of the fourth subproblem.

4. Watch http://people.cs.pitt.edu/~kirk/cs1511/BlackHoleEntropy.mp4. Write a paragraph to explain the holographic principle, and what happens to the entropy/information of objects that fall into a black hole.

Due Wednesday January 16

5. Recall the Definition of entropy of a probability distribution $X$:

$$H(X) = \sum_x P(x) \log(1/P(x))$$

Here we use the convention that capital letters are probability distributions and lower case letters are a possible outcome. So here $x$ is some possible value of the random variable $X$. So if the setting that $X$ is a random coin flip, then $x$ might either be Heads or Tails.

The purpose is of this homework is to prove a weaker version of the the upper bound portion of the source coding theorem. So assume that we have a string of $n$ letters independently drawn from a finite alphabet according to a probability distribution $X$.

(a) Assume that for all possible letters $x$, $P(x) = 1/2^k$ for some integer $k$, which may depend on $x$. So for example, for a 7 letter alphabet the probabilities could be

$$1/4, 1/4, 1/8, 1/8, 1/8, 1/16, 1/16$$
Give an binary encoding that uses in expectation exactly \( nH(x) \) bits. You must explain why your encoding uses this many bits in expectation.

Hint: Encode each letter separately.

(b) Now we drop the assumption that the probabilities have any special form. Give a binary encoding that uses in expectation at most \( n(H(x) + 1) \) bits. You must explain why your encoding uses this many bits in expectation.

Hint: Round each probability up to the next probability of the form \( 1/2^k \), and then apply the result from the previous subproblem.

Due Wednesday January 23

6. We define the conditional entropy of probability distributions \( X \) and \( Y \) as:

\[
H(X \mid Y) = \sum_y P(y) \sum_x P(x \mid y) \lg 1/P(x \mid y) = \sum_{x,y} P(x,y) \lg 1/P(x \mid y)
\]

Here \( P(x \mid y) \) is the probability of event \( x \) given event \( y \) and \( P(x,y) \) the probability of event \( x \) and event \( y \) both happening. Further define the mutual information between \( X \) and \( Y \) as:

\[
I(X;Y) = H(X) - H(X \mid Y)
\]

Intuitively, the mutual information \( I(x;y) \) measures the average reduction in uncertainty (measured in bits) about \( x \) that results from learning the value of \( y \).

(a) Now assume \( x \) is a bit that a sender wants to send to a receiver over a noisy channel, and let \( y \) be the bit received by the receiver. Because the channel is noisy, \( y \) may not equal \( x \). Let \( X \) be the probability distribution where \( x \) is 0 with probability 1/3 and \( x \) is 1 with probability 2/3. Assume the noisy channel has the following properties \( P(y = 0 \mid x = 0) = .9, \ P(y = 1 \mid x = 0) = .1, \ P(y = 0 \mid x = 1) = .2, \) and \( P(y = 1 \mid x = 1) = .8 \). Let \( Y \) be the probability distribution for the received bit.

i. What is \( H(X) \) for this example?
ii. What is the probability distribution \( Y \)? That is, what is the probability that \( y=0 \) and what is the probability that \( y=1 \)?
iii. What is \( H(Y) \) for this example?
iv. What is \( H(X \mid Y) \) for this example?
v. What is \( H(Y \mid X) \) for this example?
vi. What is \( I(X; Y) \) for this example?
vii. What is \( I(Y; X) \) for this example?

Hint: You should find that \( I(X; Y) = I(Y; X) \).

viii. Restate in plain English what it means that \( I(X; Y) = I(Y; X) \) in this setting.

(b) Now consider arbitrary probability distributions \( X \) for \( x \) and \( Y \) for \( y \).

i. Prove that \( I(X; Y) \geq 0 \).
ii. Prove \( I(X; Y) = I(Y; X) \).
FYI, Shannon’s noisy channel coding theorem says: You can get about \( \max_X I(X;Y) \) bits of information through to the receiver for each bit sent.

Due Wednesday January 23

7. Show that the following two definitions of recursively enumerable are logically equivalent:

(a) A language \( L \) is recursively enumerable iff there is a Turing machine \( M \) such that if \( x \in L \) then \( M \) accepts \( x \) and if \( x \notin L \) then \( M \) loops forever on \( x \).

(b) A language \( L \) is recursively enumerable iff there a Turing machine \( M \), with a read/write tape that is initially empty and a write-only output tape, such that only elements of \( L \) are written to the output tape, and every element of \( L \) is eventually written to the output tape.

Due Monday January 28

8. • Say that a string \( x \) of \( n \) bits is semi-incompressible if \( K(x) \geq \sqrt{n} \). Here \( K(x) \) is the Kolmogorov complexity of \( x \). Show that the set of semi-incompressible strings is not computable.

• Show that there are only finitely many incompressible strings that have the property that there are no 1 bits in the string. Recall that a string is incompressible if its Kolmogorov complexity is at least its length.

  Hint: Consider the programs \( P_1, P_2, \ldots \) where \( P_i \) outputs the \( i^{th} \) shortest string that does not contain any 1’s.

• Show that there are only finitely many incompressible strings that have the property that the number of bits that 0 in the string is equal to the number of bits that are 1 in the string. Recall that a string is incompressible if its Kolmogorov complexity is at least its length.

  Hint: You can use with out proof the fact that if you flip \( n \) fair independent coins, the probability that the number of heads is equal to the number of tails is approximately \( 1/\sqrt{n} \).

• Show that the set of incompressible strings contains no infinite subset that is recursively enumerable.

• Show that the set of compressible strings is recursively enumerable. A string is compressible if it is not incompressible.

Due Monday January 28

9. Show that there exists \( c > 0 \), such that for all strings \( x \) and \( y \), \( K(xy) < K(x) + K(y) + c \). Here \( xy \) is the concatenation of the strings \( x \) and \( y \). Here \( K(x) \) is the Kolmogorov complexity of \( x \), which is the least number of bits required by an ANSI standard C program that doesn’t read any input and outputs \( x \).

Due Wednesday January 30
10. Consider a proof rule such as:

**Proof Rule:** From the statement $\forall x \ P(x)$ one can deduce the countably infinite number of statements $P(0)$, $P(1)$, $P(2)$, $P(3)$, etc.

Somewhat more formally, you can assume that there is a Turing machine $T$ that takes as input a statement $S$, and will output a list all the statements one can conclude from $S$ using this proof rule. Here “list” means what it means within the context of the definition of recursively enumerable, $T$ will only output statements we can deduce from $S$ by the proof rule, and every statement that we can deduce from $S$ by the proof rule will eventually be listed. Prove that the following language is still recursively enumerable:

$$L = \{ S \mid \text{statement } S \text{ is provable from the finite set } A \text{ of axioms} \}$$

Hint: Now the nodes in the tree of all proofs can have infinitely many children. So breadth first search will no longer work. So you will have to find another way to search this tree so that every node is eventually reached.

Due Wednesday January 30

11. (a) Consider the following one tape Turing Machine $M$ that halts iff the number of 1’s on the input tape is odd.

- **tape alphabet** = \{0, 1, space\}
- **start state** = $q_0$
- **states** = $Q = \{ q_0, q_1, q_h \}$
- **halt state** = $q_h$. So the machine stops running if it ever reaches state $q_h$.
- **Transitions:**
  - $(q_0, 0)$ $\rightarrow$ $(q_0, 0, \text{right})$
  - $(q_0, 1)$ $\rightarrow$ $(q_1, 1, \text{right})$
  - $(q_0, \text{space})$ $\rightarrow$ $(q_0, \text{space}, \text{stay})$. So this is an infinite loop.
  - $(q_1, 0)$ $\rightarrow$ $(q_1, 1, \text{right})$
  - $(q_1, 1)$ $\rightarrow$ $(q_0, 0, \text{right})$
  - $(q_1, \text{space})$ $\rightarrow$ $(q_h, \text{space}, \text{stay})$

Show one valid computation history $H$ for the input $I = 101$ for Turing machine $M$. Note that the computation history will have the form $\#C_0\#C_1\#C_2\#\ldots\#C_k\#$ where $k$ is the number of steps that $M$ runs on $I$ and $C_i$ is the configuration of $M$ on $I$ after $i$ steps. Note that there technically there is more than one valid computation history depending on how many spaces at the right of the tape are included in each configuration.

(b) Give the Godel sentence $S$ for the $M$ and $I$ in the previous subproblem. That is, $S$ should be a first order sentence in the language of number theory that will be true iff and only if $M$ halts on $I$.

Hint: To get you started, $S$ should ask whether there exists a number $H$, which one can interpret as a computation history of $M$ on $I$. Conditions then need
to be added to $S$ to check that $H$ is a valid computation history of $M$ on $I$ that shows that $M$ halts. You are strongly encouraged to use macros (see http://www.computerhope.com/jargon/m/macro.htm), which of course you have to define.

For example, some macros that might be useful are:

- $PLACE(j)$ represents an arithmetic expression that returns the digit in location $j$ in $H$.
- $SAME(i,j,k,l)$ represents a logical expression that will be true iff digits $i$ through $j$ are identical to digits $k$ through $l$.
- $STATE(i)$ represents a local expression that will be true iff only digit $i$ of $H$ represents a state in $Q$.
- $TABLE(i,j)$ represents a local expression that will be true iff digits $i$, $i+1$ and $i+2$ in $H$ represent a tape symbol, state in $Q$ and a tape symbol respectively, and digits $j$, $j+1$ and $j+2$ in $H$ evolve properly from digits $i$, $i+1$, and $i+2$ according to $M$. So for example if digits $i$, $i+1$ and $i+2$ were $0, q_0, 1$ respectively, then $TABLE(i,j)$ would only be true iff digits $j$, $j+1$, $j+2$ were $0, 1, q_1$.

You are welcome to use other macros. But of course you have to define what they mean.

Due Friday February 1

12. Consider first order logical sentences of arithmetic where you are only allowed to use the arithmetic operators $=$ and $+$. Without loss of generality we may assume that the only logical operators are AND, represented by $\land$ and NOT, represented by $\neg$, and that all quantifiers appear first. So you might have a formula like:

$$\forall x \exists y \forall z \exists w \ (x + y = z) \land \neg (y + z = w + x)$$

Our goal here is to show that there is an algorithm to accept exactly the language of true formula. So the proof Godel’s Incompleteness Theorem doesn’t work if addition is the only allowed mathematical operation. We will use the following strategy.

Recall that we know how to build a finite state machine $L$ that accepts tuples $(x, y, z, w)$ properly encoded that satisfy $(x + y = z)$ and a finite state machine $M$ that accepts tuples $(x, y, z, w)$ properly encode that satisfy $(y + z = w + x)$.

(a) Explain how to construct a finite state machine $N$ that accepts tuples $(x, y, z, w)$ properly encoded that satisfy $\neg(y + z = w + x)$ from finite state machine $M$.

Hint: This is completely trivial. If you need more than a sentence to explain how to do this, you are not on the right track.

(b) Explain how to construct a finite state machine $P$ that accepts tuples $(x, y, z, w)$ properly encoded that satisfy $(x + y = z) \land \neg(y + z = w + x)$ from finite state machine $L$ and $N$. 
Hint: The states in $P$ will be of the form $(l, n)$ where $l$ is a state in $L$ and $n$ is a state in $N$.

(c) Now consider the quantifiers from the inside out. Explain how to construct a finite state machine $Q$ that accepts tuples $(x, y, z)$ properly encoded with the property that $\exists w \ (x + y = z) \land \neg (y + z = w + x)$ from finite state machine $P$.

Hint: The states in $Q$ will be subsets of states in $P$.

(d) Explain how to construct a finite state machine $R$ that accepts tuples $(x, y)$ properly encoded with the property that $\forall z \exists w \ (x + y = z) \land \neg (y + z = w + x)$ from finite state machine $Q$.

Hint: The states in $R$ will be subsets of states in $Q$.

(e) Note that the same idea can be used to construct a finite state machine $S$ that accepts strings $x$ properly encoded with the property that $\forall y \forall z \exists w \ (x + y = z) \land \neg (y + z = w + x)$ from finite state machine $R$.

(f) Give an algorithm that, given as input a finite automata $S$, can determine whether there is any string that $S$ accepts.

(g) Give an algorithm that decides whether a given first order sentence in number theory, that only involves arithmetic operations $+$ and $=$ and has the quantifiers appearing first, is true. More or less all the main ideas are contained in the previous subproblems. You more or less just need a couple of sentences to put everything together.

Due Monday February 11

13. For the purposes of this problem, only consider 1 tape Turing machines. In our proof of the time hierarchy theorem in class we assumed an encoding of a pair $(T, b)$, where $T$ is a Turing machine and $b$ is a positive integer, into a string. For a string $I$, let $T_I$ and $b_I$ be the Turing machine and integer encoded by $I$. We then considered the language $L$ defined to be the collection of strings $I$ such that $M_I$ on input $I$ doesn’t accept within $b_I | I |$ steps. It was more or less obvious that $L$ is not in $TIME(n)$. We then wanted to show that $L$ is in $TIME(n^c)$ for some smallish constant $c$. To do this we needed to show there exist a Turing machine $S$ and integer $a$ such that for all inputs $I$, $S$ halts on $I$ in at most $a | I |$ steps, and $S$ accepts $I$ if and only if $I \in L$.

We then considered a Turing machine $S$ that did the following on input $I$: Construct $M_I$ and $b_I$, and store them on the tape. Initialize a counter to $b_I | I |$ on the tape. Mark out $b_I | I |$ positions on the tape, and initialize them to $I$ followed by $(b_I - 1) | I |$ blanks. These positions will be used to store the work tape while simulating $T_I$. Now simulate $T_I$ on $I$ for $b_I | I |$ steps in the natural way. This simulation can be done in time less than say $100(b_I | I |)^2$ steps.

(a) What is wrong with the argument that this shows $L$ is in $TIME(n^2)$ because you can take $a = 100(b_I)^2$?

(b) Explain why $S$ can be used to show that $L$ is in $TIME(n^22^n)$? Or to ask the question in another way, why does there exist a constant $d$ such that $S$ takes at most $dn^22^n$ steps on inputs of size $n$? Note that conceptually this proves a weak
version of the time hierarchy theorem, saying that one has exponentially longer
time, one can solve more problems.

(c) So now we want to improve this to get \( \text{TIME}(n) \) is not equal to \( \text{TIME}(n^c) \) for
some smallish constant \( c \). To do this we are going to change the encoding of \( T \)
and \( b \) in \( I \). In particular, we are going to assume that \( b \) is encoded in unary in \( I \).
So if \( b = 5 \) then \( b \) is encoded as 11111, and if \( b = 8 \) then \( b \) is encoded as 11111111.
We can then proceed as before to get a language \( L' \) defined to be the collection
of strings \( I \) such that \( M_I \) on input \( I \) doesn’t accept within \( b_I \mid I \) steps (note that
the \( T_I \) extracted from \( I \) is conceptually the same as when defining \( L \), but the \( b_I \)
is different because of the different way \( b \) is encoded).

i. Explain why \( L' \) is not in \( \text{TIME}(n) \).

ii. Explain why \( L' \) is in \( \text{TIME}(n^c) \) for some smallish constant \( c \). Explicitly state
the smallest \( c \) for which your argument will work.

Due Monday February 11

14. Prove that the following two definitions of \( \text{TIME}(T(n)) \), where \( T(n) = \Omega(n) \), are
equivalent in the sense that they contain exactly the same languages.

Definition 1: \( \text{TIME}(T(n)) \) is the set of all languages \( L \) such that there exists a Turing
machine \( M \) such that (1) \( M \) accepts \( x \) iff \( x \in L \) and (2) for all but finitely many \( x \), \( M \)
on \( x \) halts in \( T(|x|) \) steps.

Definition 2: \( \text{TIME}(T(n)) \) is the set of all languages \( L \) such that there exists a Turing
machine \( N \) and a number \( b \) such that (1) \( N \) accepts \( x \) iff \( x \in L \) and (2) for all \( x \), \( N \)
on \( x \) halts within \( b * T(|x|) \) steps.

Hint: You need to show that if you have a Turing machine \( M \) that satisfies the first
condition, then you construct from it a Turing machine \( N \) that satisfies the second
condition. This direction is pretty straight-forward. And you need to show that if you
have a Turing machine \( N \) that satisfies the second condition, then you construct from
it a Turing machine \( M \) that satisfies the first condition. This direction is a bit trickier,
and you have to show how to speed up any Turing machine by a constant factor. The
main insight one needs to accomplish this is to use a larger tape alphabet size.

Due Wednesday February 13

15. (a) Let \( A \) be the language of properly nested parentheses. So for example, \( () \) and
\( ((()))() \) are in \( A \) but \( ) \) is not in \( A \). Show that \( A \) can be accepted by a log-space
Turing machine.

Hint: This is very easy. It is sufficient to store one number between 0 and \( n \) (the
number of parentheses in the input) on the work tape while making one pass over
the read-only input tape.

(b) Let \( B \) be the language of properly nested parentheses and brackets. So for example,
\( ([()])()[] \) is in \( B \), but \( [ ] \) is not in \( B \). Show that \( B \) can be accepted by a
log-space Turing machine.
Hint: I don’t see how to do this by making only one pass over the input tape. Start by making sure that \([\) and \(]\) don’t appear in the input. Then rule out substrings like \([()]\) and \(([])(\))\. Then continue in this manner.

Due Friday February 15

16. Assume a log-space reduction from a language \(A\) to a language \(B\).

So more precisely, there is a Turing machine \(T\) with three tapes, a read-only input tape, a read/write work tape, and a write-only output tape. \(T\) only uses log of the input size many cells on the read/write work tape. Further \(T\) never backs up the tape head on the write-only output tape, so the tape head on the write-only tape either stays in position or moves to the right. The machine \(T\) has the property that a string \(x\) is in \(A\) iff the contents of the write tape, when \(T\) ends computation on input \(x\), is in \(B\).

Now show that if there is a log space Turing machine \(S\) that accepts \(B\), then there is a log space Turing Machine \(U\) that accepts \(A\).

Hint: I think it’s harder to understand the question than it is to actually solve the problem once you really understand the question. The machine \(U\) has to simulate the computation of both machines \(S\) and \(T\) while only using log space. Note that this is not trivial because \(U\) does not have enough work tape to write down the output of \(T\). This is a good problem to show how to make use of the fact that space is reusable.

Due Monday February 18

17. (a) Define \(\operatorname{EXPSPACE}\) to be the set of languages \(L\) where there exists a Turing machine \(M\), and integer \(k\) such that \(M\) accepts exactly the language \(L\) and using space at most \(2^n^k\) on all inputs of size \(n\). Define a language \(C\), and show that \(C\) is complete for \(\operatorname{EXPSPACE}\) under polynomial time reductions.

   Hint: This is should be more of less line by line the same logic as showing that \(\operatorname{PSPACE}\) has a complete language, which we did in class.

   (b) Define \(\operatorname{EXPSPACE}\) to be the set of languages \(L\) where there exists a Turing machine \(M\), and integer \(k\), and an integer \(c\) such that \(M\) accepts exactly the language \(L\) and using space at most \(c^n^k\) on all inputs of size \(n\). Define a language \(C\), and show that \(C\) is complete for \(\operatorname{EXPSPACE}\) under polynomial time reductions.

   Hint: First ask yourself what issue arises here that didn’t arise in the previous subproblem. Then ask yourself how to address this issue. There are very easy ways to address this issue.

Due Monday February 18

18. Read the description of the generalized geography and the proof that it complete for \(\operatorname{PSPACE}\) under polynomial time reductions at:

   \url{https://en.wikipedia.org/wiki/Generalized_geography}

   (a) Show the Generalized Geography game instance that would result from applying the reduction to the quantified Boolean formula (you can draw this by hand, you don’t need to use LaTeX):
∃w∀x∃y∀z(x ∨ ¬y ∨ z) ∧ (w ∨ ¬y ∨ ¬z) ∧ (¬w ∨ y ∨ z) ∧ (y ∨ z ∨ ¬x)

(b) Who has a winning strategy for this instance of the Generalized Geography game, the first player or the second player? Give a brief justification for your claim.

Due Wednesday February 20

19.  (a) Problem 5.9 part (a) from the text. EXACT INDSET is defined in the chapter 5 in the text.

(b) Problem 5.11 from the text. SUCCINCT SET-COVER is defined in the chapter 5 in the text.

(c) Problem 5.13 part (a)

Hint: Most of these should be pretty straight-forward provided one understands the definition of the polynomial time hierarchy.

Due Monday February 23

20.  (a) Show that if 3SAT is polynomial time many-to-one/Karp reduction (see definition 2.7) to the complement of 3SAT then NP=co-NP. This is a simpler version of problem 5.3 from the text.

Hint: Two complexity classes are equal if they are each a subset fo the other. Understand why 3SAT and the complement of 3SAT are not obviously reducible to each other using Karp reductions. So this problem is talking about the consequence of 3SAT actually being Karp reducible to its complement.

(b) Problem 5.3 from the text.

Hint: There is nothing deep going on here. The whole issue will be understanding the definitions. Once one understands the definitions, the problem is straight-forward. I suggest reading section 5.5 first, which is < 2 pages. Then at a high level, you will want to show that if the polynomial time hierarchy collapses at some level, then it collapses at higher levels. Specifically, you want to show that if NP=co-NP then Σ^2_p = NP = co-NP and Π^2_p = NP = co-NP. Once you have that, the collapse of the higher levels follows from the repeated application of this same argument.

Due Monday February 23

21. Problem 5.9 part b and part c.

Hint: Part b is easy if you understand the definitions of NP, co-NP and DP. For part c, you can use as a black box theorem 2.15 that shows that INDSET is complete for NP. There is a conceptually simple answer. But you will need to have some understanding of the definition of DP, and what an independent set of a graph is.

Due Wednesday February 25
22. (a) Let $G$ be a Boolean formula with $n$ variables and $m$ logical operations (AND, OR, and NOT). Show how to construct a combinatorial circuit $C$ with $n$ input lines and one output line, such that $C$ outputs a 1 if and only if $G$ is true for that setting of the variables. Further the number of gates in $C$ should be $O(n^2m^2)$.

(b) Let $C$ be a combinatorial circuit with input lines $I_1, \ldots, I_n$ and one output line. Further $C$ has $S$ gates. Show how to compute a Boolean formula $G$ over variables $I_1, \ldots, I_n$, and perhaps some other variables, such that $G$ has $O(S^2)$ logical operations (AND, OR, and NOT), and $G$ is satisfiable given a particular setting of the variables $I_1, \ldots, I_n$ if and only if $C$ on those inputs, outputs a 1. So you want $G$ to essentially simulate $C$.

Hint: So the main difficulty here is that the output of one gate in a combinatorial circuit can feed into the input of many gates, which can’t really happen in a Boolean formula. So you need to figure out how to circumnavigate this issue.

(c) Problem 6.2 from the text.

Due Friday March 1

23. Problem 6.3 from the text.

Hint: Time Hierarchy Theorem. This is very easy.

Due Friday March 1

24. Show that there are Boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$ that can be computed with $n^4$ gates but not with $n^2$ gates.

Hint: Think about Shannon’s proof of Theorem 6.21 in the text. The trickiest part is probably to show that there are at least say $2^{n^3}$ different Boolean functions that can be computed with $n^4$ gates.

Monday March 4

25. Problem 6.5 from the text.

Hint: Consider the language $L$ consisting of the set of strings $x$ accepted by the lexicographically first Boolean circuit $C_{|x|}$ with $|x|^k$ gates, with the property that there is no circuit $D$ with less than $|x|^k$ gates that computes the same Boolean function as does $C_{|x|}$.

Monday March 4

26. 26. Problem 7.6 from the text. That is, show $ZPP = RP \cap \text{coRP}$

Wednesday March 6

27. Problem 7.5 from the text. Explicitly state the probability $p$ that you would use.

Hint: Use the hint at the back of the text, and the law of large numbers:


Don’t worry about making your proof fully formal, concentrate on getting the main idea. I’m mostly interested that you can correctly identify the probability $p$, and can give some intuitive reasoning how to use this probability.
28. Define the complexity class PP to be the collection of languages $L$ for which there is a polynomial $Q$ and there is a probabilistic Turing machine $M$ such that

- $M$ always runs in time at most $Q(n)$ of the input of size $n$,
- if $x \in L$ then $M$ accepts with probability strictly greater than $1/2$, and
- if $x \not\in L$ then $M$ rejects with probability strictly greater than $1/2$.

Prove $(\text{NP} \cup \text{co-NP}) \subseteq \text{PP}$.  
Hint: There is a very short proof.

Due Wednesday March 6

29. Problem 7.7 from the text.
Hint: $BP \cdot NP$ is defined in Definition 7.17. Recall proof that $BPP$ is in $P/poly$.
Due Friday March 8

30. Problem 7.9 from the text.
Hint: Recall that $BPP$ is in $\Sigma_2^p$ and proof that polynomial time hierarchy is in $\text{PSPACE}$.
Due Monday March 18

31. Problem 7.8 from the text
Hint: Recall the proof that $BPP$ is in $\Sigma_2^p$.
Due Monday March 18

32. Give a public coin interactive protocol for the language QNR defined in example 8.9 in the text.
Hint: Convert the private coin protocol given in example 8.9 into a public coin protocol using the set size protocol. Use as inspiration the conversion of the private coin GNI protocol into a public coin protocol using the set size protocol. Note here that the set $S$ that you will be concerned with is the set of possible messages sent by the verifier to the prover in the private coin protocol. You will need to use Euler’s criterion, which states that there are $(p+1)/2$ quadric residues (including 0) and $(p-1)/2$ quadratic nonresidues.
Due Friday March 22

33. Problem 8.1 from the text.
Hint: Use the fact that $\text{IP}=\text{PSPACE}$ in parts c and d.
Due Monday March 25

34. Problem 8.5 from the text.
Hint: See the hint at the back of the book. Recall the proof that $BPP \subseteq \Sigma_2^p$.
Due Monday March 25
35. Consider the interactive protocol for the language TQBF, of true quantified Boolean formulas. Consider the following formula $F$:

$$\exists x \forall y \exists z \ (x \vee y \vee \neg z) \land (\neg x \vee \neg y \vee z)$$

(a) Conceptually what is the polynomial that is used to replace $z \ (x \vee y \vee \neg z) \land (\neg x \vee \neg y \vee z)$.

(b) First consider the protocol without linearization.

i. What is the integer $S$ and polynomial $s(x)$ that Merlin sends in the first round? Show your work.

ii. Assuming that Arthur responds by sending the message $1/3$ to Merlin, what is the polynomial that Merlin sends in his second message to Arthur? Show your work.

iii. Arthur checks this second polynomial to see if it has some property, what property is this?

(c) Now consider the protocol with linearization.

i. Conceptually what is the polynomial that is used to replace $z \ (x \vee y \vee \neg z) \land (\neg x \vee \neg y \vee z)$. So you should multiply the polynomial out and linearize it.

ii. What is the integer $S$ and polynomial $s(x)$ that Merlin sends in the first round? Show your work.

iii. Assuming that Arthur responds by sending the message $1/3$ to Merlin, what is the polynomial that Merlin sends in his second message to Arthur? Show your work.

iv. Arthur checks this second polynomial to see if it has some property, what property is this?

Due Wednesday March 27

36. Show that MAM is a subset of AM. Note that this is a special case of 8.7 in the text, and that there is a hint in the book.

Further hint: Consider the MAM protocol. Assume the protocol guarantees the verifier makes the wrong decision with probability at most $1/3$. Assume the first message from the prover has $m$ bits. Explain how to modify the protocol to get another MAM protocol where the verifier is wrong with probability at most $1/4^m$. You can use the following Chernoff bound: if you have $k$ independent (unfair) coin flips that each come up heads with probability $p$, then the expect number of heads is $kp$, and the probability that the number of heads is either more than $10\%$ more or less than $10\%$ less is at most $2e^{-p^2 k/3}$. Then use the hint in the back of the book and a union bound.

Due Friday March 29

37. Read the portions of https://people.cs.pitt.edu/~kirk/cs1511/BabaiIPStory.pdf that discussed the history of the discovery of the $IP = PSPACE$ result. You
do not have do anything, but now this is fair game for a B question on the second midterm.

Due Monday April 1

38. It is shown in https://en.wikipedia.org/wiki/Fredkin_gate that the Fredkin gate is universal and reversible. Similarly show that the Toffoli gate, see https://en.wikipedia.org/wiki/Toffoli_gate for a description, is universal and reversible.

Due Monday April 1

39. Recall the half-silvered mirror experiment, which was essentially a Hadamard gate, followed by a "not", followed by a Hadamard gate. If photon came in horizontally, then came out horizontally, and if photon came in vertically, then it left vertically.

(a) Calculate the outgoing state of the photon if the incoming state is in the superposition state: \( a |H\rangle + b |V\rangle \). Here \( |H\rangle \) represents horizontal and \( |V\rangle \) represents vertical.

You can assume that
\[
|H\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]
and
\[
|V\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

You can assume that the half-silvered mirror implements the 1-bit Hadamard operation
\[
\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}
\]

You can assume that the full silvered mirror implements the not operation
\[
|V\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

(b) What is the probability that an observer sees the photon come out of the last half-silvered mirror horizontally?

(c) What is the probability that an observer sees the photon come out of the last half-silvered mirror vertically?

Due Monday April 1

40. Problem 10.6 from the text.

Due Monday April 1

41. Here we consider performing Hadamard operations on two qubits.

(a) Show the 4 by 4 matrix that represents a 1 bit Hadamard operation on the first bit.
(b) Show the 4 by 4 matrix that represents the 1 bit Hadamard operation on the second bit.

(c) Verify that the product of the two above matrices is the two bit Hadamard operation.

(d) Show the result of applying the 2 bit Hadamard operation to $a \lvert 00 \rangle + b \lvert 01 \rangle + c \lvert 10 \rangle + d \lvert 11 \rangle$.

(e) Show the result of applying the 1 bit Hadamard operation to the first qubit of $a \lvert 00 \rangle + b \lvert 01 \rangle + c \lvert 10 \rangle + d \lvert 11 \rangle$.

(f) Show the result of applying the 1 bit Hadamard operation to the first qubit $a \lvert 00 \rangle + b \lvert 01 \rangle + c \lvert 10 \rangle + d \lvert 11 \rangle$ followed by applying the 1 bit Hadamard operation to the second bit.

Hint: The fourth and the sixth answers should be the same.

Due Monday April 1

42. (a) Work out in detail the probability that Alice and Bob win in the case that $x = y = 1$ in the EPR experiment in the book. Show every step of your calculations, and give lots of explanation. You can assume without loss of generality that the order of events is Alice rotates, Bob rotates, Alice measures, and then finally Bob measures.

(b) Consider the following protocol for Alice and Bob (which I had on the board at some point).

Alice: If $x = 0$ then $a = 0$ else $a$ is the result of measuring Alice’s entangled bit after rotating it by $\pi/8$

Bob: If $y = 0$ then $b = 0$ else $b$ is the result of measuring Bob’s entangled bit after rotating it by $-\pi/8$.

Calculate the probability of Alice and Bob winning with this protocol (you can reuse calculations for the protocol in the textbook as appropriate). Is this probability more, less or the same as for the probability of Alice and Bob winning for the protocol in the textbook?

Due Wednesday April 3

43. Consider Simon’s algorithm, and the discussion of it in section 10.5 of the text. If $a = 0^n$ then $f$ is one-to-one, not two-to-one as it would be for any other value of $a$. In the case that $a = 0^n$, does Simon’s algorithm correctly compute $a$, or does the correctness of the algorithm require that $a \neq 0^n$? Justify your answer.

Due Friday April 5

44. Consider the same set up as the parity game. Alice and Bob split two entangled bits $a$ and $b$ in state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$. Alice and Bob are then split up. Alice is then given 2 classical bits $x$ and $y$. Depending on the value of $x$, and $y$, Alice performs some operations on qubit $a$ and then sends qubit $a$ to Bob. The 4 possible states of the sent qubit will be the 4 Bell states: $1/\sqrt{2} |0\rangle + 1/\sqrt{2} |1\rangle$, $1/\sqrt{2} |0\rangle - 1/\sqrt{2} |1\rangle$, $-1/\sqrt{2} |0\rangle +$
1/√2 |1⟩, and −1/√2 |0⟩ − 1/√2 |1⟩. From this single qubit, Bob determines with certainty the two classical bits x and y from his entangled bit and the sent qubit.

(a) Explain what operations Alice can perform to change the state of qubit a to each of the 4 Bell states depending on the 4 possible values of the classical bits x and y.

(b) State what the state of qubits a and b are when Bob receives them. This state will depend on the value of the classical bits x and y.

(c) Explain how Bob can retrieve the bits x and y from operations and measurements on qubits a and b. Hint: Change the basis of 4 dimensional space in which |ab⟩ live. Call this a Bell measurement.

Due Friday April 5

45. (Extra credit:) The goal of this problem is to find a way to transmit information about a qubit by sending two classical bits. Alice and Bob split up entangled bits a and b in state |00⟩/√2 + |11⟩/√2. Assume that now Alice is given qubit x. So x is in some unknown superposition α |0⟩ + β |1⟩ between states |0⟩ and |1⟩. Alice and Bob then follow the following informally stated protocol:

- Alice performs a Bell measurement on qubit a and qubit x, which you developed in the last homework problem. This results in 4 possible outcomes.
- Alice sends Bob two classical bits describing the outcome of this measurement.
- As a result of Alice’s measurement, qubit b is in one of four possible states. Of these four possible states, one is identical to the original quantum state, and the other three are closely related. Which of these four possibilities actually obtains is encoded in the two classical bits. Knowing this, the qubit b can be modified in one of three ways, or not at all, to result in a qubit identical to x.

Explain how to implement the last step, and prove that it works. To prove it works you just need to show by calculations that the final state of qubit b is the initial state of qubit x.

Alice now performs the following reversible operation on x, if a = 1 then negate x. Alice then runs qubit a through a Hadamard gate. Alice now measures the current values of a and x, and sends these two classical bits to Bob. Explain what the state of all the particles a, b, and x is after each of Alice’s operations. Then explain how Bob can use the two classical particles to change the state of b to the original state of x. Note that when Alice measures a and x, then she can no longer recover the original state of qubit x.

Due Monday April 8

46. Problem 10.5 from the text

Due Monday April 8
47. Read Scott Aaronson’s intuition for Shor’s quantum polynomial time factoring algorithm, found here: http://www.scottaaronson.com/blog/?p=208. You don’t have to do anything. But this is now fair game for a B question on midterm 2.

Due Monday April 8

48. Problem 11.2 from the text. This is very easy.

Due Wednesday April 10

49. Problem 11.7 from the text. This should be pretty straightforward. Just convert the interactive proof for the permanent from chapter 8 as the interactive proof for graph non-isomorphism was converted in example 11.7.

Due Wednesday April 10

50. Problem 11.16 from the text. Note that “equation” in this context means that the relation is equality. So one possible equation might be $1.5x + 2y + 2.5z = 10.5$. Note that rational solutions are allowed. So $x = 2, y = .5$ and $z = 3$ is an allowable solution.

Hint: There is a not too complicated reduction from MAXSAT.

Due Wednesday April 10

51. (a) Give a solution $u$ to the following instance of QUADEQ:

\[
\begin{align*}
  u_1u_1 + u_1u_2 + u_3u_3 & = 0 \\
  u_1u_3 + u_2u_2 & = 1 \\
  u_1u_1 + u_1u_2 & = 1 \\
  u_2u_2 + u_2u_3 & = 0 \\
  u_2u_3 + u_1u_3 + u_1u_2 & = 1
\end{align*}
\]

That is give a $u_1, u_2,$ and $u_3$ that satisfy these equations.

(b) Write a program in Java, or your favorite programming language if you don’t like Java, that takes as input a solution $u$ to some instance of QUADEQ and produces the probabilistically checkable proof constructed in in the proof of Theorem 11.19 in the text (and the proof we did in class). Turn in your code. You code can assume that $u$ is not more than 5 bits. Input and output conventions don’t really matter. Turn in your code.

(c) Give the output of your program for the $u$ you found in the first subproblem. Hint: You output should have 520 bits.

(d) Specify which of these 520 bits would give you the value of: $u_1u_2 + u_2u_2 + u_3u_3$. So I’m looking for a number between 1 and 520 inclusive. Show your work, as the answer may depend upon the convention on ordering the bits.

Friday April 12

52. Problem 11.9 from the text

Friday April 12
53. Problem 9.3 from the text
   Hint: This is super easy. There is more or less a one line proof.
   Due Monday April 15

54. Problem 9.5 from the text
   Hint: Show that the problem of inverting a one-way function can be solved by a non-
deterministic polynomial-time machine. You may assume (without any real loss of
generality) that for all inputs, the output of the one way function contains at least as
many bits as the input.
   Hint: This is super easy. There is more or less a one line proof.
   Due Monday April 15

55. Problem 9.2 from the text
   Hint: First do the proof for $m = 2$ and $n = 1$. It will be seem way easier, but the
general proof is more or less the same as the proof when $m = 2$ and $n = 1$.
   Due Monday April 15

56. Problem 9.11 from the text
   Hint: This is relatively straightforward.
   Due Monday April 15

57. Problem 9.6 from the text.
   Hint: Part (a) is easy using a trick we have seen before. For part (b) show how to use
a polynomial time algorithm for inverting $f_U$ to build a polynomial time algorithm for
inverting $f$.
   Due Wednesday April 17

58. Read and understand the protocol in problem exercise 9.17 part b from the text.
   (a) Write a one paragraph summary of the protocol in your own words.
   (b) Then write a paragraph explaining intuitively what it means for an interactive
   proof to be computationally zero knowledge.
   (c) Then write a paragraph to explain why it is intuitively believable why this protocol
   is computationally zero knowledge.
   You do not have to formally prove anything.
   Due Wednesday April 17