HW1
Due: September 15, 2016

Attach a fully filled-in cover sheet to the front of your printed homework. Your name should not appear anywhere except on the cover sheet; each individual page of the homework should include your Banner ID instead.

While collaboration is encouraged in this class, please remember not to take away notes from any collaboration sessions. Also please list the names and logins of any of your collaborators at the beginning of each homework.

Please monitor Piazza, as we will post clarifications of questions there. You should hand in your solutions by 12:55 to the CSCI 1010 bin on the second floor of the CIT. Late homeworks are not accepted.

Problem 1

Use induction to prove that for any positive integer \( n \), \( n^3 + 2n \) is divisible by 3.

Problem 2

**Conjunctive Normal Form** (CNF) is a particular way of writing Boolean formulae that will be used in this course. We define it constructively:

1. A *literal* is a variable or its complement. \( x_i \) and \( \neg x_i \) are examples of literals.

2. A *clause* is any disjunction (that is, OR) of any number of literals. A literal by itself is also a clause.

3. A formula is *in conjunctive normal form* if it is the conjunction (that is, AND) of any number of clauses.

As an example, the formula \( (x_1 \lor x_3) \land (\neg x_1 \lor x_2) \land \neg x_2 \) is in CNF.

Using the above definition, convert the following formulae into conjunctive normal form:

a. \( (x_1 \land x_2 \land x_3) \lor (\neg x_1 \land x_4) \lor \neg x_2 \)
b. \( \neg(x_1 \lor (x_2 \land x_3)) \lor x_2 \)

c. \( (x_1 \leftrightarrow x_2) \leftrightarrow x_3 \)

d. \( (x_1 \rightarrow x_2) \oplus (x_2 \rightarrow x_3) \)

Recall that \( \oplus \) represents exclusive or where \( x_1 \oplus x_2 = (x_1 \lor x_2) \land \neg(x_1 \land x_2) \), \( \rightarrow \) represents implies where \( x_1 \rightarrow x_2 = \neg x_1 \lor x_2 \), and \( \leftrightarrow \) represents if-and-only-if where \( x_1 \leftrightarrow x_2 = (x_1 \rightarrow x_2) \land (x_2 \rightarrow x_1) \).

**Problem 3**

The Bega and O’Nion families have been feuding for a long time. They can’t stand to share things, and want all of their houses to be unique. Unfortunately, there’s a suburb of Myami, a small agricultural community, in which the only houses belong to one of the two families. Since they like unique things, every member wants to have a unique number of neighbors, and a fight seems likely if someone from the other family forces a number to be repeated.

The police are worried, and are pretty sure that a peaceful solution is impossible. They have hired you, an aspiring mathematician, to prove it for them. In other words:

Suppose \( G = (V, E) \) is an undirected graph such that \( |V| \geq 2 \). Prove that at least two vertices must have equal degree.

**The following questions are lab problems.**

**Lab Problem 1**

1. When it was first discovered that there are computational problems that cannot be solved by computers, it was a very surprising result. In this problem, we will do a high-level argument of a related result, due to Gödel. Gödel showed that there exist true mathematical statements that cannot be proven. Instead of using mathematical statements, we will simply use English sentences.
This is the sentence we are trying to prove:

*There exists a true sentence for which there is no proof.*

We will argue this by contradiction.¹ That is, we will assume the following statement, then obtain a contradiction:

*Assumption: All true sentences have proofs.*

You may assume that any statement for which there exists a proof is true. (Hint: Examine the sentence \( S = \text{“No proof exists for this sentence.”} \) Show that given our assumption, if \( S \) is true or false, we obtain a contradiction.)

2. In class we looked at the Halting Problem, the problem of determining whether a program \( P \) will terminate on an input \( X \). We wanted to know whether a program \( \text{HPSolver} \) that solves the Halting Problem exists:

\[
\text{HPSolver:}
\]
\[
\text{Input: a program } P, \text{ an input } X \\
\text{Output: } Yes \text{ if } P \text{ terminates on } X, \text{ No if } P \text{ runs forever}
\]

We concluded that \( \text{HPSolver} \) cannot exist, meaning that no program can solve the halting problem. We will now look at another problem that cannot be solved by computers, the problem of deciding the truth or falsity of a sentence. We want to know whether a program \( \text{TruthSolver} \) exists:

\[
\text{TruthSolver:}
\]
\[
\text{Input: a sentence } S \\
\text{Output: } True \text{ if } S \text{ is true, False if } S \text{ is false}
\]

Argue that \( \text{TruthSolver} \) cannot exist, by first showing that if we had a \( \text{TruthSolver} \), we could use it to build an \( \text{HPSolver} \). Then argue that since \( \text{HPSolver} \) cannot exist, \( \text{TruthSolver} \) cannot exist. (Hint: \( \text{HPSolver} \) will use \( \text{TruthSolver} \) as a subroutine. What kind of statement should it give as input?)

¹We are being informal here, because we have not given a mathematical formalization of the statement we are trying to prove.
Lab Problem 2

Mealy vs. Moore  Moore machines are finite state machines (FSMs) where an output/action is produced when the FSM reaches a state. That is to say, each transition is labeled by some input symbol, and each state is labeled with an output/action.

In a Mealy machine an output/action is produced as the FSM makes a transition from one state to another, not after it reaches a state. That is, each transition is labeled with a pair \(<\text{input}>/\text{action}>\), and the action is performed as the machine transitions from one state to another. In part a, the possible actions are \(\text{yam}\) and \(\text{potato}\).

a) Given this Mealy machine, give us an equivalent Moore machine, one that will produce the same action/output on the same input.

b) Given this Moore machine, give us an equivalent Mealy machine, one that will produce the same action/output on the same input.
c) It turns out that, in general, for every Mealy machine there is an equivalent Moore machine, and vice versa. You just saw an example of this equivalence. Based on the way you solved (a) and (b), explain a systematic way of constructing a corresponding Moore machine from a Mealy machine, and vice versa.

Lab Problem 3

Let $G$ be a graph with $n$ nodes where $n \geq 3$. For each pair of nodes $(u, v) \in G$, there is a directed edge either $u \rightarrow v$ or $v \rightarrow u$. Prove that $G$ has a node $x$ called the übertüber such that there is a directed path with length at most two from every node in $G$ to the übertüber.

Hint: There is an inductive proof that has the base case of a graph with three nodes. Consider all situations that could arise when you add in a fourth node.