Problem 1

a. $2n = \mathcal{O}(n) \because c \cdot n \geq 2n$ when $c = 2$

b. $n^2 \neq \mathcal{O}(n) \because$ no c exists such that $c \cdot n \geq n^2$ for arbitary values of n

c. $n^2 = \mathcal{O}\left(n \log^2 n\right) \because \mathcal{O}\left(n \cdot \log^2 n \cdot \log^2 n\right) = \mathcal{O}(n \cdot n) = \mathcal{O}\left(n^2\right)$ Assuming $\log_2$

d. $n \log n = \mathcal{O}\left(n^2\right) \because \mathcal{O}\left(n^2\right) = \mathcal{O}(n \cdot n) \wedge \log n \leq n$

e. $3^n = 2^{\mathcal{O}(n)} \because 3^n = 2^{n \cdot \log_2 3}$ Notice then $\log_2 3$ is a constant

f. $2^{2^n} = \mathcal{O}\left(2^{2^n}\right) \because c \cdot 2^{2^n} \geq 2^{2^n}$ when $c = 1$

Problem 2

a. $n = o(2n) \because \lim\limits_{n \to \infty} \frac{n}{2n} = \frac{1}{2}$

b. $2n = o(n^2) \because \lim\limits_{n \to \infty} \frac{2n}{n^2} = 0$

c. $2^n = o(3^n) \because \lim\limits_{n \to \infty} \frac{2^n}{3^n} = 0$

d. $1 = o(n) \because \lim\limits_{n \to \infty} \frac{1}{n} = 0$

e. $n \neq o(\log n) \because \lim\limits_{n \to \infty} \frac{n}{\log n} = \infty$

f. $1 \neq o(\frac{1}{n}) \because \lim\limits_{n \to \infty} \frac{1}{1/n} = \infty$

Problem 3

1. $1274 \mod 10505 = 1274$
2. $10505 \mod 1274 = 313$
3. $1274 \mod 313 = 22$
4. $313 \mod 22 = 5$
5. $22 \mod 5 = 2$
6. $5 \mod 2 = 1$

1274 and 10505 are relatively prime

1. $7289 \mod 8029 = 7289$
2. $8029 \mod 7289 = 740$
3. $7289 \mod 740 = 629$
4. $740 \mod 629 = 111$
5. $629 \mod 111 = 74$
6. $111 \mod 74 = 37$

7289 and 8029 are not relatively prime

Problem 4

x	y	$(x \vee y) \wedge (x \vee \sim y) \wedge (\sim x \vee y) \wedge (\sim x \vee \sim y)$
0	0	0
0	1	0
1	0	0
1	1	0

No combinations produce a one, so the formula is not satisfiable

Problem 5

Union. Let L_1, L_2 be languages in P. M_1 and M_2 accept L_1 and L_2 respectively. We can construct a machine, M_0, which has two tapes. M_0 simulates M_1 on the first tape and M_2 on the second. Run the input to M_0 on the first tape. If the machine accepts in polynomial time, accept. Otherwise, run the input on the second tape. If the machine accepts, accept. Otherwise reject.

Complement. Suppose M is a machine which accepts the languge L in polynomial time. We can use M to construct M'. M' simulates M. If M accepts in polynomial time, then M' rejects. If M rejects, then M' accepts.

Concatanation. Suppose M_1, M_2 accept L_1 and L_2 in polynomial time. We can construct M_0 which maintains a counter, i. We simulate M_1 and M_2 on M_0. For the input, s, we run the sub-string $s_0, s_1 \ldots, s_i$ on M_1. If M_1 rejects, we increment i and repeat. If M_1 accepts, we run the remaining input on M_2. If M_2 accepts, accept, otherwise reject.

Problem 6

Union. Let L_1, L_2 be languages in NP. By definition, We can construct two polynomial time verifiers for L_1 and L_2. We can therefore construct a verifier, V_0, for $L_1 \cup L_2$. This verifier returns true if either V_1 or V_2 return true. Because bothe V_1 and V_2 run in polynomial time, V_0 also runs in polynomial time. This means that $L_1 \cup L_2$ is in NP.

Concatenation. Let V_1 and V_2 be polynomial time verifiers for the languages L_1 and L_2 in NP. We can construct another polynomial time verifier which verifies $L_1 L_2$ by splitting our input at i. We run a substring of the input from the start to i, $s_0, s_1, \ldots, s_i$ on V_1, which runs in polynomial time. If V_1 does not verify, then increment i and repeat. If V_1 does verify, run the remaining input on V_2. Because both V_1 and V_2 run in polynomial time, this verifier is polynomial, so the concatanation of two NP languages is also NP.

Problem 7

In the worst case, this algorithm must iterate over each unmarked node once for each loop where we mark a node (to mark additional nodes). We safely excude marked nodes from this loop. This means we need to perform $|v| + (|v| - 1) + (|v - 2|) + \cdots + 0 = \sum_{i=0}^{|v|} |v| - i$ operations. This is clearly less than n^2 operations, so this algorith is $\mathcal{O}\left(n^2\right)$ which is polynomial.

Problem 8

For a DFA to accept Σ^*, every accept state which can be reached from our initial state, q_0, must be accepting. If we perform a depth-first search on the states on our DFA rooted at q_0, we will have to check a maximum of $|Q|$ states. If we encounter a non-accepting state, we reject. The complexity of this algorithm is therefore $\mathcal{O}(|Q|)$. This is a polynomial function, so this languge must be in P.