Problem 1

To show that SUBGRAPH ISOMORPHISM NP-complete, we must show that it is in NP and is NP-hard. We can see that SUBGRAPH ISOMORPHISM is in NP because we can simply go through all vertices and edges in G and H to confirm the mapping between vertices. This occurs in polynomial time.

Now, to show that SUBGRAPH ISOMORPHISM is NP-hard, we can reduce it from CLIQUE which we know is NP-complete. Let us consider some instance of CLIQUE, (G, k). We must show a reduction from (G, k) to an instance of SUBGRAPH ISOMORPHISM, (G, H). We can let G contain n vertices. It is important to note that if $k > n$ then we produce a 'no' instance because we cannot create a clique of more vertices than the number in the graph. Otherwise, we can create a k clique using this k. This process occurs in polynomial time because to generate a k clique, we simply iterate over the possible edges to connect which occurs in polynomial time. Additionally, once we have our generated clique, H, we simply copy over G which also occurs in polynomial time. Now we must show that both yes maps to yes and no maps to no for this reduction.

To show that yes maps to yes, let's suppose that we have some 'yes' instance of CLIQUE. It follows that G has a subgraph which is isomorphic to a k clique. Thus, if we let H be this k clique, we have shown that G contains an isomorphic subgraph to H. Thus, we have shown that yes maps to yes.

To show that no maps to no, we can show that if our instance (G, H) was in SUBGRAPH ISOMORPHISM, then the (G, k) that it was reduced from is in CLIQUE. We know that H is a k clique and G has a subgraph isopmorphic to H. Thus, it follows that G has an isomorphic subgraph to a k clique. Thus, $(G, k) \in$ CLIQUE. Thus, we have shown that no maps to no.

Since we have shown a valid reduction from CLIQUE to SUBGRAPH ISOMORPHISM, it follows that SUBGRAPH ISOMORPHISM is NP-complete.

Problem 2

To show that SET COVER NP-complete, we must show that it is in NP and is NP-hard. We can see that SET COVER is in NP because we can simply go through all elements to check if they occur in at least one set. This process occurs in polynomial time.

Now, to show that SET COVER is NP-hard, we can reduce it from VERTEX COVER which we know is NP-complete. Consider an instance of VERTEX COVER, (G, k), where G is an undirected graph and k is a positive integer. We can construct an instance of SET COVER, (C, k'), where $C = S_1, S_2, ..., S_n$, $U = \cup_i S_i$, and $k' = k$. We can show that if there exists a vertex cover of size at most k in G, then there exists a set cover of size at most k' in C. Suppose we have a vertex cover, V', of G with $|V'| \leq k$. Then for each vertex $v_i \in V'$, we can select the corresponding set $S_i \in C$. Since V' is a vertex cover of G, every edge in G is connected to at least one vertex in V'. Thus, it follows that every edge in C is included in at least one set in the selected cover. Thus, the selected sets form a valid cover for C and $|T| = |V'| \leq k$. Now, we can show that if there exists a set cover of size at most k' in C, then there exists a vertex cover of size at most k in G. Let T be a set cover for C where $|T| \leq k'$. For each set $S_i \in T$, we can choose some arbitrary vertex $v_i \in G$ that corresponding to the edges incident to v_i. Since $|T| < k'$, we know that T vertices are selected in G. Also, since each edge in C is covered by T, every edge in G is connected to at least one selected vertex. Thus, the selected vertices form a valid vertex cover for G, and $|V'| = |T| \leq k$. It is clear to see that reduction is in polynomial time. Now we must show that both yes maps to yes and no maps to no for this reduction.

To show that yes maps to yes, let us consider an instance of VERTEX COVER, (G, k). It follows that there exists a vertex cover V' of size at most k and the SET COVER instance, (C, k'), has a cover T of size $\leq k$. We know this since either endpoint of the edge (u, v) is contained in C so there exists an $S_i \in C$. Thus, we have shown that yes maps to yes.

To show that no maps to no, let us consider an instance of SET COVER, (C, k'), that maps to 'yes'. It follows that there exists a cover T of size $\leq k'$, and the corresponding vertices in G form a vertex cover V' of size $\leq k$. Here we notice that for every edge (u, v), u, v exist in some S_u and S_v which implies that either S_u or S_v is in T.

Problem 3

To show that MIN BISECTION NP-complete, we must show that it is in NP and is NP-hard. We can see that MIN BISECTION is in NP because we can simply iterate through S to check that the number of nodes is $\frac{n}{2}$. Also, we can iterate through the edges to verify that there are at least k edges that cross from S to V/S. This process occurs in polynomial time.

Now, to show that MIN BISECTION is NP-hard, we can reduce it from MAXCUT which we know is NP-complete. Let us consider some instance of MAXCUT, $G = \{(V, E), k\}$ and $|V| = n$. We can add n isolated nodes to G which does not change the bisection problem (since isolated nodes do not have any edges). These n nodes allows us now to make any cut into bisection. For example, we can consider a cut on the original G which splits the vertices into V and V/S but such that $|V| \neq |V/S|$. Now, in the new graph, we can add $|V/S|$ isolated nodes to V's partition and $|V|$ nodes to V/S's partition. Now let's consider the complement of the new graph, G'. G' has $2n$ nodes and contains every edge not in G. Since this is a multigraph, we must consider the parallel edges. Let us consider two vertices (u, v) which have p parallel edges between them. Let m be the largest number of parallel edges between any two vertices. Then, there would be $m - p$ parallel edges between (u, v) in G'. Thus, we have defined the complement of our multigraph. Thus, by reducing MAXCUT to MAX BISECTION to MIN BISECTION, we have shown that G has a bisection with at least k edges if and only if $mn^2 - k$ cross the bisection in its complement. This process occurs in polynomial time. Now we must show that both yes maps to yes and no maps to no for this reduction.

To show that yes maps to yes, let's suppose that we have a bisection in G with k edges crossing. It follows that there are at most $mn^2 - k$ edges that cross in G'. Thus, we have shown that yes maps to yes.

To show that no maps to no, let's suppose that there are at most $mn^2 - k$ edges that cross a bijection in G'. It follows that at least K edges cross the cut in G. Thus, we have shown that yes maps to yes.

Since we have shown a valid reduction from MAXCUT to MIN BISECTION, it follows that MIN BISECTION is NP-complete.

Problem 4

<u>a.</u>

To show that PARTITION NP-complete, we must show that it is in NP and is NP-hard. We can see that PARTITION is in NP because given a subset, we can iterate through each partition and check if the sums of the partitions are equal. This process occurs in polynomial time.

Now, to show that PARTITION is NP-hard, we can reduce it from SUBSET - SUM which we know is NP-complete. Let us consider an instance of SUBSET - SUM, $(a_1, a_2, ..., a_n, B)$. Let $S = \sum_{i=1}^{n} a_i$. We can let $(a_1, a_2, ...a_n) \cup \{S - 2B\}$ be what we feed into PARTITION. This process occurs in polynomial time. Now we must show that both yes maps to yes and no maps to no for this reduction.

To show that yes maps to yes, let's suppose there exists a subset in the instance of SUBSET-SUM that sums to B. Then, we know that the complementary set sums to $S - B$. If this is true, it follows that in $(a_1, a_2, ...a_n) \cup \{S - 2B\}$, there exists a partition such that the sum of each partition equals $S - B$. Thus we have shown that yes maps to yes.

To show that no maps to no, lets suppose there exists a partition of $(a_1, a_2, ..., a_n) \cup \{S - 2B\}$ such that the sums of both partitions equal $S - B$. It follows that one of the sets must contain the number $S - 2B$. If we remove this value from the set, we get $S - B - (S - 2B) = B$, which was our goal. This is our original SUBSET - SUM instance. Thus, we have shown that no maps to no.

Since we have shown a valid reduction from SUBSET - SUM to PARTITION, it follows that PARTITION is NP-complete.

b.

To show that KNAPSACK NP-complete, we must show that it is in NP and is NP-hard. We can see that KNAPSACK is in NP because we can simply iterate through the values to make sure the total value is at least V and at most C. This process occurs in polynomial time.

Now, to show that KNAPSACK is NP-hard, we can reduce it from SUBSET - SUM which we know is NP-complete. Let us consider an instance of SUBSET - SUM, $(a_1, a_2, ..., a_n, B)$. We can simply set the budget constraint and the minimum value equal to each other such that $a_i = c_i = v_i$. This process clearly occurs in polynomial time. Now we must show that both yes maps to yes and no maps to no for this reduction.

To show that yes maps to yes, let's suppose there exists a subset, $T \subseteq S$ such that $a_{a \in T} a = B$. It follows that since $V = C = B$, if this set were a knapsack, it will have a budget $C = B$ and value $V = B$. Thus we have shown that yes maps to yes.

To show that no maps to no, lets suppose there exists a knapsack $\{(c_1, c_2, ...c_n, v_1, v_2, ..., v_n, V, C) : \exists T \subseteq \{1, 2, ...n\}$ for which $\sum_{t \in T} v_t \geq Vand\sum_{t \in T} c_t \leq C\}$. Then, it follows by construction $V = C = B$, since $a_i = c_i = v_i$. Therefore $\sum_{t \in T} v_t \geq V = B$ and $\sum_{t \in T} c_t \leq C = B \rightarrow \sum a_i = B$. Thus, we have shown that no maps to no.

Since we have shown a valid reduction from SUBSET - SUM to KNAPSACK, it follows that KNAPSACK is NP-complete.