1. We are asked to show that if the bin packing decision problem (BP-Decision) can be solved in polynomial time, then bin packing optimization problem (BP-Optimization) can also be solved in polynomial time.

   Assume BP-Decision $\in P$ and consider the following algorithm:

   ```
   BP-Optimization($S$)
   for $k = 1$ to $|S|$
     if BP-Decision($S, k$) == true
       return $k$
     end if
   end for
   ```

   The above algorithm calls the decision problem BP-Decision as a subroutine. The algorithm takes as input a set S, and returns an integer k that is the smallest number of bins required to hold all of the objects in S.

   The largest number of bins is clearly $|S|$, where every object is in a different bin. Therefore we simply walk from $k = 1$ to $k = |S|$ calling our decision problem each time to determine whether or not we can pack the objects into k bins. The running time of BP-Optimization is $O(n \cdot O(\text{BP-Decision}))$. Therefore if BP-Decision $\in P$, then BP-Optimization can be solved in polynomial time. □

2. First we note that we only need to find which edges are specifically necessary for a hamiltonian cycle. Once we find that minimum set of necessary edges, we can pick any vertex as a starting point, and the order of the vertices will be defined by the set of edges.

   Assume HC-Decision $\in P$ and consider the following algorithm:

   ```
   HC-Order(G)
   G' = G
   for all edges e in E
     remove e from G'
     if HC-Decision(G') == true
       permanently remove e from G'
     end if
   endfor
   return G'
   ```

   In the above algorithm, we walk all of the edges, checking that condition each time, and at the end of the algorithm, we are left with a graph that contains only the hamiltonian cycle. Therefore, we have the order. The running time of HC-Order is $O(|E| \cdot O(\text{HC-Decision}))$. Therefore, if HC-Decision $\in P$, then HC-Order can be solved in polynomial time. □

3. (a) Let TSP(G,k) be a Boolean function that solves the traveling salesperson decision problem for a weighted graph $G = (V, E)$ and cycle weight bound k.

   Let W be the sum of the weights of all edges. The weight of an optimal cycle is at most W. The idea of the algorithm is to use the binary search to find the optimal cycle weight.

```
TSP-Optimization(G)
low = 1
high = W
while low < high
  mid = (low+high)/2
  if TSP(G,mid)
    high = mid - 1
  else
    low = mid + 1
  end if
endwhile
```

   The number of iterations of the loop is $\Theta(\log W)$. Since $W \leq |E| \times (\max\{\text{largest edge weight}\})$, $\log W$ is polynomially bounded in the size of the input. If TSP runs in polynomial time, then so the whole algorithm does.

   (b) From (a), we can find the weight of an optimal tour. Let us refer it as w_{opt}. Now we can use the the Boolean function TSP(G, w_{opt}) again by probling the edges as we do in Problem 2 to an optimal tour.

4. (a) Here is a $O(V + E)$ algorithm for testing if a graph is 2-colorable. Modify BFS to alternate coloring of the nodes. Basically, we are checking for a cycle with an odd length of edges. If a node is colored red, all of its adjacent neighbors must be colored blue. If a node ever has an adjacent neighbor that has already been visited, its color must be opposite the color of the current node. If BFS reports an odd length cycle, the graph is not 2-colorable.

   (b)

   1) First we show that 4-Color $\in$ NP.
      Given a graph G, and a coloring assignment of the vertices, simply walk the graph and make certain that all adjacent vertices have a different color, and make certain that only 4 colors are used. This is clearly $O(V + E)$.

   2) Next we show that 3-Color $\leq_T$ 4-Color.
      Let G^3 be an instance of 3-Color. Construct a new graph G^4 as follows: Add a single extra vertex v and connect it to every other vertex in the graph. This is clearly polynomial in the size of the graph.
      Now we must show that

      G^3 is a *yes*-instance of 3-Color **iff** G^4 is a *yes*-instance of 4-Color:

      "$\Longrightarrow$" Assume G^3 is 3-colorable. Therefore, G^4 is 4-colorable because the added vertex v, which is connected to all the other vertices in the graph, can be colored with a 4th color, and it will always be connected to vertices that are 1 of 3 other colors.
      "$\Longleftarrow$" Assume G^4 is 4-colorable. Because v is connected to every vertex in the graph, v must be the only vertex in G^4 that has a certain color. Therefore, all other vertices in the graph are colored 1 of 3 colors. Therefore, G^3 is 3-colorable.

   Since we have shown that 4-Color $\in$ NP and 3-Color $\leq_T$ 4-Color, then we hae shown that 4-Color $\in$ NP-Complete. $\square$

5. Recall that the SUBSET-SUM problem states as follows:

   > Given a positive integer c, and the set $S = \{s_1, s_2, \dots, s_n\}$ of positive integers s_i for $i = 1, 2, \dots, n$. Assume that $\sum_{i=1}^{n} s_i \geq c$. Is there a $J \subseteq \{1, 2, \dots, n\}$ such that $\sum_{i \in J} s_i = c$?

   An instance of SUBSET-SUM problem is a set of numbers and a target c. It is known that SUBSET-SUM problem is NP-complete.

   To show that the Set-Partition problem is NP-complete, we need to show that it can be reduced from SUBSET-SUM.

   1. We verify that Set-Partition is in NP.
      Given an instance of Set-Partition, we use the subset A as a certifcate. We can easily check whether the set A is a correct partition by summing the values in A and the values in $S - A$, and comparing them. This certainly takes a polynomial time. Therefore, Set-Partition is in NP.
   2. To show Set-Partition $\in$ NP-Hard, we reduce from Subset-Sum to Set-Partition, that is to show that

      $$\text{Subset-Sum} \leq_T \text{Set-Partition}.$$

      To do so, let S be an instance of Subset-Sum with the target c. Let us define the transformation by constructing a new set of numbers S' as follows:

      $$S' = S \cup \{u, v\},$$

      where

      $$u = 2w - c, \quad \text{and} \quad v = w + c, \quad \text{and} \quad w = \sum_{x \in S} x$$

      Since we are only adding two elements to the set S, this is clearly a polynomial time transformation.
      Now we must show that

      $$S \text{ is a Yes-instance of Subset-Sum} \iff S' \text{ is a Yes-instance of Set-Partition.}$$

      $\Longrightarrow$ Let $J \subseteq S$ and the elements in J sum to c. Therefore, $J \cup \{u\}$ sum to $2w$ because $c + u = c + 2w - c = 2w$. Note that the elements in $\overline{J} = S - J$ sum to $w - c$. Hence, $\overline{J} \cup \{v\}$ also sums to $2w$ because $w - c + v = w - c + w + c = 2w$. Therefore, S' can be partioned into $J \cup \{u\}$ and $\overline{J} \cup \{v\}$ where both partitions sum to $2w$. Therefore, a Yes-instance of Subset-Sum transforms to a Yes-instance of Set-Partition.

      $\Longleftarrow$ Assume S' can be partitioned into two sets, A and $\overline{A} = S' - A$, such that

      $$\sum_{x \in A} x = \sum_{x \in \overline{A}} x \tag{1}$$

      Since $w + u + v = w + 2w - c + w + c = 4w$, the sum of the elements in both sets must be equal to $2w$. Furthermore, u must be in one set and v must be in the other because $u + v = 2w - c + w + c = 3w$. Without loss of generality, let $u \in A$, then by (1),

      $$2w = \sum_{x \in A} x = u + \sum_{x \in A - u} x = 2w - c + \sum_{x \in A - u} x,$$

      or equivalently,

      $$\sum_{x \in A - u} x = c$$

      Thus, $J = A - u$ is a subset of elements that sum to c. A Yes-instance of Set-Partition transforms to a Yes-instance of Subset-Sum. $\square$

3. It is known that Subset-Sum $\in$ NPC. We have shown that Subset-Sum $\leq_T$ Set-Partition. Therefore, we conclude that Set-Partition $\in$ NP-Complete.

Alternative answer of Problem 5:

1. First, we verify that Set-Partition is in NP. Given an instance of Set-Partition, we use the subset A as a certifcate. We can easily check whether the set A is a correct partition by summing the values in A and the values in $S - A$, and comparing them. This certainly takes a polynomial time.
2. To show the Set-Partition is NP-hard, we show that

   $$\text{Subset-Sum} \leq_T \text{Set-Partition}$$

   To do so, let S be an instance of Subset-Sum with the target c. Define the instance of Set-Partition by

   $$S' = S \cup \{y\},$$

   where

   $$y = 2c - \sum_{x \in S} x,$$

   The sum of the numbers of $\mathcal{S}'$ is $2c$.

   Now we show that

   $$\text{Yes-instance of Subset-Sum} \iff \text{Yes-instance of Set-Partition}$$

   $\Longrightarrow$ Assume we know some subset J of S whose elements total c, i.e., $\sum_{x \in J} x = c$. Then we can take a partition of $\mathcal{S}'$ by taking $A = J$. Then we can see that

   $$\sum_{x \in A} x = \sum_{x \in J} x = c.$$

   On the other hand,

   $$\sum_{x \in \bar{A}} x = \sum_{x \in S' - J} x = \sum_{x \in S'} x - \sum_{x \in J} x = 2c - c = c.$$

   Therefore, we obtain a Yes-instance of Set-Partition.

   $\Leftarrow$ Suppose S' can be partitioned into two subsets whose sums are equal. Since the sum of all elements in S' is $2c$, the sum of each partition must be c. Furthermore, since we have a partition of S', the element y must be in exactly one of the partitions; the other partition must therefore be a subset of S, yielding a subset of S whose elements add up to exactly c.
3. It is known that Subset-Sum $\in$ NPC. We have shown that Subset-Sum $\leq_T$ Set-Partition. Therefore, we conclude that Set-Partition $\in$ NP-Complete.