Question one

Problem: Longest Path Length
Input: A weighted graph \( G = (V, E, W) \) and two nodes \( u \) and \( v \).
Output: The length of the longest simple path between \( u \) and \( v \).

a) Transform this problem into a decision problem.

Problem: Longest Path Decision
Input: A weighted graph \( G = (V, E, W) \), two nodes \( u \) and \( v \), and a bound \( k \).
Output: True if the graph \( G \) has a simple path between \( u \) and \( v \) of length \( k \) or greater, false otherwise.

\[ \text{L-PATH} = \{ <G, u, v, k> | \text{Graph } G \text{ has a simple path between } u \text{ and } v \text{ of length } k \text{ or greater} \} \]

b) Prove that Longest Path Decision (L-PATH) and Longest Path Length are reducible to each other.

// Solve decision with general
L-PATH(G, u, v, k)
    if Longest-Path-Length(G, u, v) >= k
        return true
    return false

The if statement is constant time and so are the return statements, so the reduction is constant time and therefore polynomial time (\( \Theta(n^0) \)).

// Solve general with decision
Longest-Path-Length(G, u, v)
    for k = G.numVertices to 0 do
        if L-PATH(G, u, v, k) == true
            return k
    return "no path"

The for loop is order \( \Theta(v) \) where \( v = |G.V| \), which is polynomial time in relation to the input length. \( u \) and \( v \) are constant length, and \( G \)'s length is determined by both its edges and vertices. However there are at most \( v^2 \) edges and at least 0. With 0 edges, \( G \) is just a list of vertices, and the running time is \( \Theta(v) \). With \( v^2 \) edges the input is \( \Theta(v + v^2) = \Theta(v^2) \) and the running time is still \( \Theta(v) \), so this is clearly polynomial time with an exponent of \( \frac{1}{2} \).
Question two

Problem: Clique
Input: An undirected graph G and a value k.
Output: "Yes" if G has a complete subgraph of size ≥ k and "No" otherwise.

Prove Clique is in NP by following the steps below.

a) What is the certificate for the Clique problem?
A set of ≥ k vertices in graph G that are all connected.

b) What is the verification algorithm?

// Input: Undirected graph G, clique lower bound k, and answer subset a which // will be verified // Output: True if a is a clique of size >=k, false otherwise
Verify-Clique(G, k, a)
if a.size < k
    return false
for each vertex v1 in a
    for each vertex v2 in a
        if v1!=v2 and there is no edge v1-v2 in G
            return false
return true

The first if statement is constant time, as is the return statement. That leaves the double for loop, which both loop over n elements where n is the size of G. Inside the for loops are some more constant time statements.

Therefore the running time is c + n(n(c)), which is Θ(n^2). This is polynomial time since it is of the form n^m where m = 2 and is a constant.
c) *Show that Largest Clique is reducible to Decision Clique and vice versa.*

// Solve general with decision
//
// Input: Graph G
// Output: Largest clique in G
Largest-Clique(G)
   for k = G.numVertices to 0
      if Decision-Clique(G,k) == true
         return size

The for loop runs $\Theta(v)$ where $v = |G.V|$, which is polynomial time in relation to the input. See problem 1b relating $|v|$ to $|G|$.

// Solve decision with general
//
// Input: Graph G and bound k
// Output: True if a clique of size $\geq k$ exists in G, false otherwise
Decision-Clique(G,k)
   if Largest-Clique(G) $\geq$ k
      return true
   return false

If and return statements are constant/polynomial time (see 1b).
Question three

Demonstrate the following by giving poly-time algorithms to solve $C$.

a) If $A$ and $B$ are in $P$, then the problem $C = \{x : x$ is an instance of both $A$ and $B\}$ is also in $P$.

\[
C(x) \\
\text{if } (A(x)==true) \text{ and } (B(x)==true) \\
\text{return true} \\
\text{return false}
\]

If and return statements are constant/polynomial time (see 1b).

b) If $A$ and $B$ are in $P$, then the problem $C = \{x : x$ is an instance of either $A$ or $B\}$ is also in $P$.

\[
C(x) \\
\text{if } (A(x)==true) \text{ or } (B(x)==true) \\
\text{return true} \\
\text{return false}
\]

If and return statements are constant/polynomial time (see 1b).

c) If $A$ and $B$ are in $P$, then the problem $C = \{z : where z = xy for some x and y where x is an instance of A and y is an instance of B\}$ is also in $P$.

\[
C(z) \\
\text{for } (i=0 \text{ to } z\text{.length}) \\
\text{if } A(z\text{.substring}(0,i))==true \text{ and } B(z\text{.substring}(i,z\text{.length}))==true \\
\text{return true} \\
\text{return false}
\]

The for loop is $\Theta(|z|)$, which is exactly the length of the input, so the running time is linear and therefore polynomial with exponent one.