1. **Solution:** For the sake of contradiction, suppose there is an algorithm DECIDEL374H that correctly decides the language L_{374H}. Then we can solve the halting problem as follows:

```
DECIDEHALT(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on input w
        return TRUE
  if DECIDEL374H(⟨M′⟩)
      return TRUE
  else
      return FALSE
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.
Then M' halts on and accepts *every* input string x.
Therefore, M' halts on at least 374 distinct input strings.
So DECIDEL374H accepts the encoding $\langle M' \rangle$.
So DECIDEHALT correctly accepts the encoding $\langle M, w \rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.
Then M' diverges (i.e., does not halt) on *every* input string x.
Therefore, M' does not halt on at least 374 distinct input strings.
So DECIDEL374H rejects the encoding $\langle M' \rangle$.
So DECIDEHALT correctly rejects the encoding $\langle M, w \rangle$.

In both cases, DECIDEHALT is correct, that is, there exists a decider for HALT. But that is impossible, because HALT is undecidable. We conclude that the algorithm DECIDEL374H does not exist, i.e., L_{374H} is undecidable. ■

Rubric: Out of 10 points:

- 2pt: correct reduction direction (knowing to use decider of L_{374H} to construct algorithm for an undecidable language)
- 4pt: reduction construction (clear description of M' using M, w)
- 4pt: proof of correctness, 2pt for each direction.

Note: Wrong reduction direction is automatic zero.

2. **Solution:** To prove that EIGHT in NP-complete, we have to prove that it is in NP and that it is NP-hard.

(a) We begin by proving that EIGHT is in NP. We let the certificate be a pair of lists of vertices, each of length $k+1$, where each list corresponds to the vertices of one of the cycles in order, and the both lists begin and end with the vertex the two cycles have in common.

The following is then a valid certifier:

```
EightCertifier(⟨G, k⟩, ⟨L_1, L_2⟩):
  if length(L_1) ≠ k + 1 OR length(L_2) ≠ k + 1:
      REJECT
  v = L_1[0]
  if L_1[k] ≠ v OR L_2[0] ≠ v OR L_2[k] ≠ v:
      REJECT
  visited = {v}
  for i from 1 to 2:
      if {v, L_i[1]} ∉ E(G):
          REJECT
      for j from 1 to k − 1:
          if L_i[j] ∈ visited:
              REJECT
          if {L_i[j], L_i[j + 1]} ∉ E(G):
              REJECT
          visited = visited ∪{L_i[j]}
  ACCEPT
```

If $\langle G, k\rangle$ does have an eight-graph on $2k-1$ vertices, passing the vertices of the two cycle of the eight graph in order, beginning with the common vertex will cause the certifier to pass all of it's checks. On the other hand, if if the certifier accepts, it means that the certificate contained two distinct cycles (since we checked that the vertices were distinct and that each vertex had an edge to the next one in the list) which share exactly one vertex (v), and this corresponds exactly to the definition of an eight-graph.

Furthermore our certificate for "yes" instances is polynomial length (it is a list of vertices, so its length is linear), and our certifier runs in polynomial time (the outer look runs twice, the inner one runs k times), the other lines are either array accesses, checking set membership, and adding to a set, all fo which can certainly be done in at most linear time.

Thus, since we have created an efficient certifier for it, EIGHT is in NP.

(b) Now we show that EIGHT is NP hard by reducing HAMILTONIAN-CYCLE to EIGHT. Since we know that HAMILTONIAN-CYCLE is NP-Hard, this will show that EIGHT is NP-Hard. To do this reduction, we need describe a polytime function f mapping graphs to pairs of graphs and integers such that if $f(G) = (G', k)$ then G' has a $2k-1$ vertex eight-graph if and only if G has a Hamiltonian cycle.

Given an n-vertex graph G, let v be an arbitrary vertex of G, and let G' be the graph which is the same as G but with $n-1$ new degree 2 vertices added to form a cycle containing v. Let $k = n = |V(G)|$

We claim that G' contains a $2n-1$ eight-graph if and only if G has a Hamiltonian cycle.

If G has a Hamiltonian cycle, then we can form an eight-graph in G' by simply taking the Hamiltonian cycle in G along with the n vertex cycle containing v. These two cycles both have length n, and share the single vertex v.

On the other hand, if G' has a $2n-1$ vertex eight-graph as a subgraph, this eight graph must use all of vertices of G'. Each of the newly added vertices is only in one n-cycle, the one containing all of the new vertices along with v, so this must be one of the n-cycles of the eight-graph. Thus, the other n-cycle of the subgraph must use all the vertices from G, meaning that it is a Hamiltonian cycle for G, and thus G has a Hamiltonian cycle.

Finally, f can be computed in polynomial time, since we are simply adding a linear number of vertices to G.

■

Rubric: Out of 10 points:

- 4pts: EIGHTS is in NP
  - 1pt: Describes a correct format for certificates
  - 1pt: Attempts to construct a certifier
  - 1pt: Certifier is correct
  - 1pt: Justifies correctness of certifier
- 6pts: EIGHTS is NP-Hard
  - 1pt: Attempts to convert instances of some NP-hard problem to instances of EIGHTS/Attempts to solve an NP-hard problem using a black box for EIGHTS (zero points for NP-hardness if reduction is in the wrong direction)
  - 2.5pts: Reduction is correct
  - 2pts: Justify reduction correctness (1 pt for each direction)
  - 0.5pt: Justify reduction is polytime

Note: Wrong reduction direction is automatic zero.

3. **Solution:** (a) **CLIQUE-COVER to SAT:** The input to the CLIQUE-COVER problem is the graph $G(V, E)$ and integer k, while that to the SAT problem is a boolean formula ϕ. We reduce CLIQUE-COVER to SAT as follows. For every vertex u in V, we define k variables $x(u, i)$, $\forall i \in [k]$. A variable being 1 in a satisfying assignment of SAT will imply that the corresponding node is in the partition i in the CLIQUE-COVER instance.

   A solution to the CLIQUE-COVER problem must satisfy some criteria. To have the corresponding SAT solution reflect these criteria, we define different types of clauses, one for each criterion, for the SAT instance as follows.

   - Non-adjacent vertices cannot belong in the same clique: For every non-edge $(u, v) \notin E$, define k clauses: $(\overline{x(v, i)} \vee \overline{x(u, i)})$, $\forall i \in [k]$. That is, variables corresponding to two non-adjacent vertices and the same partition cannot both be 1.
   - Every vertex should belong in exactly one partition: We define the clauses: $(\vee_{i \in [k]} x_{u,i}) \wedge (\wedge_{i,j \in [k]} (\overline{x_{u,i}} \vee \overline{x_{u,j}}))$. That is, exactly one of the $x(u, i)$s can be 1 in every satisfying assignment. Note that, if all variables $x_{u,i}, i \in [k]$ are 0, then the first clause (OR of all variables) will be false, while if two or more variables are 1, the clause corresponding to the OR of the complement of the pair will be false. As a satisfying assignment must satisfy every clause, every solution will enforce exactly one variable is 1.

   The boolean formula ϕ is an AND of all the above clauses.

   If the graph has n vertices, ϕ is a formula of exactly nk variables. Also ϕ has at most kn^2 clauses, of size 2 each, of the first type, with k clauses for every non-edge, and

kn clauses, of size k each, of the second type. Thus, the total number of clauses is at most $O(kn^2)$, which is polynomial in the size of the CLIQUE-COVER instance.

By reducing to SAT, a known NP-Complete problem, we have proved CLIQUE-COVER is in NP. However, this reduction does not prove CLIQUE-COVER is NP-Complete, as this requires proving NP-Hardness too. For doing the same, a reduction from a known NP-Complete problem to CLIQUE-COVER must be shown.

(b) **GRAPH COLORING to CLIQUE-COVER:** We prove CLIQUE-COVER is NP-Hard by reducing GRAPH-COLORING, a known NP-Complete problem, to CLIQUE-COVER. This reduction will prove CLIQUE-COVER is NP-Hard. Combined with the proof of membership in NP from part (a), this will prove CLIQUE-COVER is NP-Complete.

The reduction is as follows. The input to GRAPH-COLORING is a graph $G(V, E)$, and an integer k.

We define the complement of the graph $G(V, E)$, denoted by $\overline{G}(V, \overline{E})$, as the graph where the vertex set is the same, and the edges are all non-edges of G, that is $(u, v) \in E \iff (u, v) \notin \overline{E}$.

We claim that G **has a proper coloring using at most** k **colors if and only if** $\overline{G}$ **has a Clique-cover of size at most** k**.**

$\Leftarrow$ Note that all vertices colored by one color in a proper coloring of G form an independent set (no pair of vertices are adjacent). Hence, this set will form a clique (every pair of vertices is adjacent) in $\overline{G}$. Hence, if G can be colored using k colors, there will be k cliques in $\overline{G}$.

$\Rightarrow$ Given a clique cover of size k, we color all vertices in the same clique V_i using color i. Every pair of vertices in V_i is connected in $\overline{G}$, hence disconnected in G. Hence, coloring them using the same color does not violate coloring constraints.

$\overline{G}$ can clearly be constructed in time polynomial in the size of G, hence this is a polynomial time reduction.

■

Rubric: 5 points for part (a):

(a) -5: incorrect/wrong approach

(b) -1: variable set of SAT incorrect

(c) -1: clauses for verifying clique condition not defined/ incorrectly defined

(d) -1: clauses for 'vertex in exactly one clique' condition not defined/ incorrectly defined

(e) -1: polynomial size analysis not shown

(f) -1: not NP-Complete answer not included/ is incorrect

5 points for part (b):

(a) -5: incorrect/wrong approach

(b) -2: reduction (chosing an NP-Complete problem, mapping inputs of both problems) incorrect

(c) -1: $\leftarrow$ proof incorrect/ missing

(d) -1: $\rightarrow$ proof incorrect/ missing

(e) -1: polynomial size analysis not shown