1. Let \( \text{Dead-Code} = \text{DC} = \{\langle Q \rangle \mid Q \text{ contains a line of code that is not executed on any input to } Q\} \).

Suppose that you are given an algorithm \( A_{\text{DC}} \) that decides \( \text{DC} \). Using \( A_{\text{DC}} \) as a subroutine, give an algorithm \( A_{\text{H}} \) to decide \( \text{H} \).

2. Let \( \text{Even-Length-Halt} = \text{ELH} = \{\langle Q \rangle \mid Q \text{ eventually halts on all input strings of even length}\} \).

Suppose that you are given an algorithm \( A_{\text{ELH}} \) that decides \( \text{ELH} \). Using \( A_{\text{ELH}} \) as a subroutine, give an algorithm \( A_{\text{EQ}} \) to decide \( \text{EQ} \). Recall that \( \text{EQ} \) is the set of all pairs of programs that halt on exactly the same input:

\[ \text{EQ} = \{\langle Q_1, Q_2 \rangle \mid \text{for all inputs } x, Q_1(x) \text{ eventually halts exactly if } Q_2(x) \text{ eventually halts}\} \].

3. Let \( \text{Inf-Mixed} = \text{IM} = \{\langle Q \rangle \mid Q \text{ eventually halts on infinitely many inputs and fails to halt on infinitely many inputs}\} \).

Suppose that you are given an algorithm \( A_{\text{IM}} \) that decides \( \text{IM} \). Using \( A_{\text{IM}} \) as a subroutine, give an algorithm \( A_{\text{H}} \) to decide \( \text{H} \), or give an algorithm to decide some other undecidable language encountered in Chapter 4.

4. (a) Let \( \text{Non Tautology} \) be the following problem.

Input: A DNF formula \( F \) (i.e., a boolean formula which is an “or” of clauses, where each clause is an “and” of boolean variables and their complements).

Question: Does \( F \) have a non-satisfying assignment, that is an assignment of truth values to its Boolean variables that causes the formula to evaluate to \( \text{FALSE} \)?

E.g. \( F_1 = (x_1) \lor (x_2) \) has the non-satisfying assignment \( x_1 = \text{FALSE}, x_2 = \text{FALSE} \); \( F_2 = (x_1 \land x_2) \lor (\bar{x}_1 \land \bar{x}_2) \) has the non-satisfying assignment \( x_1 = \text{FALSE}, x_2 = \text{TRUE} \).

Show that \( \text{Non Tautology} \) has a polynomial time verifier. That is, given a candidate certificate \( C \) of size polynomial in \( n \), where \( n \) is the input size, there is a polynomial time algorithm to check whether \( C \) certifies that the input \( F \) is in the language \( \text{Non Tautology} \), and furthermore, for each \( F \in \text{Non Tautology} \), there is a polynomial-sized certificate \( C \) for \( F \).

(b) Let \( \text{Subset Sum} \) be the following problem.

Input: A collection of \( n \) not necessarily distinct positive integers and a target integer \( t \).

Question: Is there a subset of the collection, such that the numbers in the subset sum to exactly \( t \)?

E.g. Let the collection of integers be \( \{1, 2, 2, 6\} \) and the target be 5. The subset adding up to the target is 1,2,2.

Show that \( \text{Subset Sum} \) has a polynomial time verifier.
(c) The \textit{Traveling Peddler} (TS) is the following problem. 
Input: A directed graph $G = (V,E)$, where each edge has a non-negative integer length, and an integer $b$.
Question: Does $G$ have a Hamiltonian Circuit of length at most $b$?
Show that \textit{Traveling Peddler} has a polynomial time verifier.

(d) Let \textit{Composites} be the following problem.
Input: An $n$-bit integer $m$.
Question: Is $m$ a composite, that is a non-trivial product of two integers; in other words are there integers $r$ and $s$, with $r, s \neq 1$, where $m = rs$?
Show that \textit{Composites} has a polynomial time verifier.
Comment. Interestingly, there is a polynomial time algorithm (called a \textit{primality test}) to test this property; however this algorithm does this without identifying a pair $r$ and $s$ of divisors for $m$.

5. (a) Suppose that you were given a polynomial time algorithm for Directed Hamiltonian Path (DHP). Using it as a subroutine, give a polynomial time algorithm for Undirected Hamiltonian Path (UHP). (you saw the opposite reduction in class).

(b) Suppose that you were given a polynomial time algorithm for Undirected Hamiltonian Path (UHP). Using it as a subroutine, give a polynomial time algorithm for Undirected Hamiltonian Circuit (UHC).

(c) Suppose that you were given a polynomial time algorithm for Traveling Peddler. Using it as a subroutine, give a polynomial time algorithm for Directed Hamiltonian Circuit.

(d) Let \textit{Equal Subset Sum} be the following problem.
Input: A collection of $n$ not necessarily distinct positive integers, $a_1, a_2, \ldots, a_n$.
Question: Is there a subset of the collection, such that the numbers in the subset sum to exactly $\frac{1}{2} \sum_{i=1}^{n} a_i$?
Suppose that you were given a polynomial time algorithm for \textit{Equal Subset Sum}. Use it as a subroutine to give a polynomial time algorithm for \textit{Subset Sum}. 

2