1. We saw earlier that any two disjoint co-r.e. sets could be separated by a recursive set. Show that there are recursively inseparable r.e. sets, i.e. two disjoint r.e. sets \( A \) and \( B \) such that for any recursive set \( C \) with \( A \subseteq C \) we must have \( C \cap B \neq \emptyset \).

[Hint: Try \( A = \{ e : \varphi_e(e) = 0 \} \) and \( B = \{ e : \varphi_e(e) = 1 \} \).]

2. Recall the definition:

\[
A \oplus B = \{ 2n : n \in A \} \cup \{ 2n + 1 : n \in B \}
\]

Define

\[
\bigoplus_{i=1}^n A_i = \bigcup_{i=1}^n \{ nx + i - 1 : x \in A_i \}
\]

Show that \( \bigoplus_{i=1}^n A_i \) has the same Turing degree as

\[
(\cdots (A_1 \oplus A_2) \oplus \cdots) \oplus A_n
\]

3. A Non-deterministic Post Algorithm (NPA) is a finite set of pairs \( (x, y) \) which operate on a string as follows: Given a pair \( (x, y) \) and a word \( w \in A^* \), if \( x \) occurs at the beginning of \( w \) then we remove it and append \( y \) to the end, i.e. if \( w = xu \), then we replace it by \( w' = uy \). The algorithm behaves similarly to a Non-deterministic Markov Algorithm in all other respects. Show that a set \( B \subseteq \mathbb{N} \) is r.e. if and only if there is an NPA \( M \) such that for all \( w \), \( w \in B \) iff \( M \) accepts \( w \).

[Note: The difference between an NPA and an NMA is similar to the difference between a queue and a stack.]

4. A graph on \( \mathbb{N} \) is a (countably infinite) graph whose vertex set is the set of natural numbers. Such a graph is said to be recursive if the edge relation is recursive, i.e. the function

\[
E(n, m) = \begin{cases} 
1 & \text{if } n \text{ and } m \text{ are connected by an edge} \\
0 & \text{if they are not connected}
\end{cases}
\]

is a total recursive function. Show that the following problem is undecidable: Given two recursive graphs on \( \mathbb{N} \), determine if they are isomorphic, i.e. if there is a bijection of \( \mathbb{N} \) with itself such that two natural numbers are connected by an edge in the first graph if and only if their images are connected by an edge in the second graph.

[Hint: It suffice to do the following: Find some fixed graph \( G \) and an effective map \( M \mapsto G_M \) from codes for Turing machine to recursive graphs on \( \mathbb{N} \) (i.e. edge functions as defined above) such that Turing machine \( M \) halts on empty input iff \( G_M \) is not isomorphic to \( M \). The idea (very roughly) is to make \( G_M \) have some infinite connected component in case \( M \) never halts, and have only finitely many edges if it does, and let \( G \) have an infinite connected component.]