1. Three statements are given below. What can you deduce about their truth values using only the fact that the halting problem is undecidable, plus the related basic definitions? (Thus, for each statement the alternatives are “true”, “false” and “cannot be deduced from the given information.”)

   (a) There is an algorithm that for any given program \( P \) can decide the inputs on which \( P \) will halt.
   (b) There is a program \( P \) such that no algorithm can decide the inputs on which \( P \) will halt.
   (c) For all programs \( P \) it is the case that no algorithm can decide the inputs on which \( P \) will halt.

2. Define the language 

   \[ ALL_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^* \} \]

   We assume that the DFA can be encoded in a similar way as the Turing machines can be encoded. Show that \( ALL_{DFA} \) is decidable.

3. [Sipser Exercise 3.4] Give a formal definition of an enumerator (see page 154). Consider it to be a type of two-tape Turing machine that uses its second tape as the printer. Include a definition of the enumerated language.

4. Consider enumerators, as in Problem 3. Show the following.

   (a) A language is Turing-recognisable if and only if some enumerator enumerates it.
   (b) A language is decidable if and only if some enumerator enumerates it in the order of length starting from the shortest strings.
   (c) A language \( L \) is decidable if and only if both \( L \) and its complement \( L \) can be enumerated by an enumerator.

5. [Part of Sipser Problems 3.15 and 3.16]

   (a) Show that the class of decidable languages is closed under union, intersection and complementation.
   (b) Show that the class of recognisable languages is closed under union and intersection. Why can’t you use the same construction as in part (a) for complementation?

6. We are given the task of implementing a tool for program checking. According to the specification, the user can load a Java program into the tool. After that, the user can choose one line of code in the Java program, and give an input. The tool should tell whether the Java program with the given input would eventually execute the chosen line of code.

   Can such a tool be implemented? Justify your answer.