1. (80pt) For each of the following languages, prove, without using Rice's Theorem, whether it is (i) in D, (ii) in SD but not in D, or (iii) not in SD.

   1 $L_1 = \{<M>| \; \{\varepsilon, \texttt{ab}, \texttt{abab}\} \subseteq L(M)\}$

   2 $L_2 = \{<M>| \; L(M) \cap (\texttt{ab})^* \text{ is infinite}\}$

   3 $L_3 = \{<M>| \; L(M) \cap (\texttt{ab})^* \text{ is finite}\}$

   4 $L_4 = \{<M>| \; L(M) \cap (\texttt{ab})^* = \emptyset\}$

   5 $L_5 = \{<M>| \; L(M) \cap (\texttt{ab})^* \neq \emptyset\}$

   6 $L_6 = \{<M>| \; L(M) \neq L(M') \text{ for any other TM } M'\}$

   7 $L_7 = \{<M> \mid \neg L(M) \in D\}$.

   8 $L_8 = \{<M> \mid L(M) \in \text{SD}\}$.

1. In SD, but not in D

Proof that in SD:

Construct M1:

- For each w of ($\epsilon, ab, abab$¿ :
    - Run M on w
    - If M accepts w, continue looping to next w. Else, loop but do not iterate w.

M1 will halt & accept if ($\epsilon, ab, abab$¿ $\subseteq L(M)$. Therefore, SD.

Proof that not in D – Reduction from H:

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:
    a. Erase the tape.
    b. Write w on the tape.
    c. Run M on w.
    d. Accept.
2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) decides L_1:

- < M, w> $\in$ H: M halts on w, so M# accepts all inputs, including a. Oracle accepts.
- < M, w> $\notin$ H: M does not halt on w, so M# accepts nothing. Oracle does not accept

But no machine to decide H can exist, so neither does Oracle.

2. Not in SD

Proof – Reduction from $\neg$ H:

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:
    a. Save x (input)
    b. Erase tape
    c. Write w on the tape.
    d. Run M on w for |x| steps
        i. If M halts, then loop
    e. Accept. (If M doesn't halt)
2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) semidecides $\neg$ H assuming it semidecides L L:

- < M, w> $\in$ $\neg$ H: M does not halt on w, so M# accepts all inputs. Therefore M# accepts infinitely many string $\rightarrow$ Oracle accepts.
- < M, w> $\notin$ $\neg$ H: M halts on w, so M# only accepts string within a finite set $\rightarrow$ Oracle does not accept

Since $\neg$ H is not semidecidable, this is a contradiction therefore L is not SD.

Since L is not SD, it is not D.

3. Not in SD

Proof – Reduction from ¬H:

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:

   f. Save x (input)
   g. Erase tape
   h. Write w on the tape.
   i. Run M on w
   j. Accept

2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) semidecides ¬H assuming it semidecides L:

- < M, w> ∈ ¬H: M does not halt on w; M# does not accept any string. Therefore M accepts finitely many strings → Oracle accepts.
- < M, w> ∉ ¬H: M halts on w, so M# accepts every string → M is infinite -> Oracle does not accept

Since ¬H is not semidecidable, this is a contradiction → therefore L is not SD.

Since L is not SD, it is not D.

4. Not in SD

Proof – Reduction from ¬ H:

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:

    k. Save x (input)
    l. Erase tape
    m. Write w on the tape.
    n. Run M on w
    o. Accept

2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) semidecides ¬ H assuming it semidecides L:

- < M, w> ∈ ¬ H: M does not halt on w; M# does not accept any string. Therefore M accepts finitely many strings → Oracle accepts.
- < M, w> ∉ ¬ H: M halts on w, so M# accepts every string → M is infinite -> Oracle does not accept

Since ¬ H is not semidecidable, this is contradiction → therefore L is not SD.

Since L is not SD, it is not D.

5. In SD but not in D

Proof of SD by construction of M1 using following algorithm:

1. Use dovetailing to run M on every string that comes from (ab)*
2. If M accepts, then accept

M1 will halt and accept if string is in L. Therefore L is in SD.

Proof of not D – Reduction from H:

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:
    - p. Erase the tape.
    - q. Write w on the tape.
    - r. Run M on w.
    - s. Accept.

2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) decides L:

- < M, w> $\in$ H: M halts on w, so M# accepts all inputs, including a. Oracle accepts.
- < M, w> $\notin$ H: M does not halt on w, so M# accepts nothing. Oracle does not accept

But no machine to decide H can exist, so neither does Oracle.

6. L is in D, and therefore also in SD. We know this since a language that is semidecided by any turing machine is also semidecided by an infinite number of other turing machines, therefore L ¿ $\varnothing$, which is obviously decidable.

7. Not SD.

Proof – Reduction from $\neg H$

Let R(<M, w>) =

1. Construct the description <M#> of a new Turing machine M#(x) that, on input x, operates as follows:

    1.1 Save x (input)

    1.2 Erase tape

    1.3 Write w on the tape.

    1.4 Run M on w

    1.5 re-put x on tape

    1.6 If encoding of x (<M',w'>) is incorrect: Reject

    1.7 Run M' on w'

    1.8 Accept

2. Return <M#>.

If Oracle exists, then C = Oracle(R(<M, w>)) semidecides $\neg$ H assuming it semidecides L:

- < M, w> $\in$ $\neg$ H: M does not halt on w and doesn't make it past step 1.4. Therefore M# accepts $\varnothing$ → since $\neg\varnothing$ is equal to $\Sigma^{¿}$ which is not in D, Oracle accepts.
- < M, w> $\notin$ $\neg$ H: M halts on w, M# accepts encoding <M',w'> only if M' halts on w'. Therefore, H = L(M#). since $\neg H\, is\, not \in D\, , Oracle\, does\, not\, acceept$

Since $\neg$ H is not semidecidable, this is contradiction → therefore L is not SD.

8. L consists of all correct encodings of TMs because L(M) $\in$ SD is always true for any M. **Therefore L is in D.**

Part 2 –

2. (20pt) For each of the languages in question 1, indicate whether Rice's Theorem can be used or not to prove that the corresponding language is not in D. Explain why.

1. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L1 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

2. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L2 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

3. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L3 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

4. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L4 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

5. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L5 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

6. Cannot be used because L6 is in D, and Rice's theorem can only prove if something is not in D but not that it is in D. (Somewhat similar to how pumping lemma can only prove if a language is not regular, but cannot prove if its regular.)

7. Can be used. A correct application of rice's theorem must identify a non-trivial class of turing-recognizable classes. Since L7 is a non-trivial class of Turing-recognizable languages (1 turing-recognizable language is a non-trivial class of Turing-recognizable languages.), therefore rice's theorem can be used.

8. Cannot be used because L8 is in D, and Rice's theorem can only prove if something is not in D but not that it is in D. (Somewhat similar to how pumping lemma can only prove if a language is not regular, but cannot prove if its regular.)