Exercise [16.17]

ASSERTION:

There is no Turing Machine **D** that can decide for all $t, s \in N$ if $\mathbf{T}_t(N) = \mathbf{T}_s(N)$ (wherein $\mathbf{T}_t$ and $\mathbf{T}_s$ are Turing Machines producing recursive sets).

PROOF:

For purposes of illustration, I first depict an exemplary table comprising all Turing Machines ordered by their index *t* (with # being the symbol if the Turing Machine does not stop):

		1	**2**	**3**	**…**	**n**	**…**
$\mathbf{T}_1$	**1**	0	0	1	…	$\mathbf{T}_1$(n)	…
$\mathbf{T}_2$	**2**	5	1	1	…	0	…
$\mathbf{T}_3$	**3**	5	7	11	…	1	…
$\mathbf{T}_4$	**4**	0	#	#	…	1	…
…	**…**	…	…	…	…	…	…
$\mathbf{T}_t$	**t**	1	1	0	…	$\mathbf{T}_t$(n)	…
…	**…**	…	…	…	…	…	…
$\mathbf{T}_x$	**x**	12	#	34	…	$\mathbf{T}_x$(n)	…
$\mathbf{Q}_x$	**…**	0 (→ copy)	0 (→ copy)	1 (→ invert)	…	$\mathbf{Q}_x$(n)	…
…	**…**	…	…	…	…	…	…
$\mathbf{T}_s$	**s**	1	1	1	…	$\mathbf{T}_s$(n)	…
…	**…**	…	…	…	…	…	…
Y	**…**	23	55	2	…	**Y**(n)	…
…	**…**	…	…	…	…	…	…
E ??	**…**	…	…	…	…	…	…
…	**…**	…	…	…	…	…	…
D ??	**…**	…	…	…	…	…	…
…	**…**	…	…	…	…	…	…

1. Consider a given Turing Machine $\mathbf{T}_t$ that generates a first recursive set.
2. For an arbitrary Turing Machine $\mathbf{T}_x$, consider the associated Turing Machine $\mathbf{Q}_x$ which determines if $\mathbf{T}_x$ has stopped after *k* steps or not:

$$Q_x(k) = \begin{cases} 1 & \text{if } T_x(k) \text{ has stopped after } k \text{ steps} \\ 0 & \text{if } T_x(k) \text{ has NOT stopped after } k \text{ steps} \end{cases}$$

3. Now define a Turing Machine $\mathbf{T}_s$ which produces a new recursive set by:

$$T_s(n) = \mathrm{mod}_2(T_t(n) + Q_x(n)) = \begin{cases} T_t(n) & \text{if } Q_x(n) = 0 \\ \mathrm{mod}_2(T_t(n)+1) & \text{if } Q_x(n) = 1 \end{cases}$$

In Words: $\mathbf{T}_s$ copies $\mathbf{T}_t$ for each *n* for which the arbitrary Turing Machine $\mathbf{T}_x$ does not stop after *n* steps and inverts it for the other *n*'s (see green arrows in the table).

4. Identity of the first recursive set and the new recursive set (formally: $\mathbf{T}_t(N) = \mathbf{T}_s(N)$) is thus given iff $\mathbf{Q}_x(k) = 0$ for all *k*, or $\mathbf{Q}_x = 0$ for short.

5. Accordingly, existence of the desired Turing Machine **D** would imply that there exits a Turing Machine, say **E**, which can generally (for all *x*) decide if $\mathbf{Q}_x = 0$ (Proof: Apply **D** to the recursive sets produced by $\mathbf{T}_t$ and $\mathbf{T}_s$; if they are the same, $\mathbf{Q}_x = 0$; otherwise not).

6. (Probably there are smarter ways to continue, but I only found this argument:)

   Existence of **E** would imply that a Turing Machine can be constructed which can generally decide if an arbitrary Turing Machine **Y** will stop when applied to an arbitrary *n* (which is contradiction to the statement expressed in Penrose's book at the end of the preceding paragraph; hence **E** and **D** cannot exist, q.e.d.):

   Consider the Turing Machine $\mathbf{Y}^*$ defined by

$$Y^*(k) = Y(n) \quad \forall\, k \in \mathrm{N}$$

   ($\mathbf{Y}^*$ simply has the value **Y***(n)* for all inputs). Applying the hypothetic Turing Machine **E** to $\mathbf{Y}^*$ would then allow to decide if the computation of **Y***(n)* stops ($\Rightarrow Q_{Y^*} \neq 0$) or not ($\Rightarrow Q_{Y^*} = 0$).