<u>Theorem</u> A language L is decidable $\iff$ L is Turing-recognizable and co-Turing-recognizable.

<u>Proof</u> "$\Longrightarrow$" If L is decidable $\Rightarrow$ L is Turing-recognizable. Note that if L is decidable $\Rightarrow$ $\exists$ a Turing machine M that decides L. Build a Turing machine $\tilde{M}$ that reverses the output of M, i.e. if M accepts a string w, then $\tilde{M}$ rejects the same string w. If M rejects w, then $\tilde{M}$ accepts w. $\tilde{M}$ is therefore a decider for $\overline{L}$ $\Rightarrow$ $\overline{L}$ is Turing-decidable $\Rightarrow$ $\overline{L}$ is Turing-recognizable; so L is Turing-recognizable and co-Turing-recognizable.

"$\Longleftarrow$" If both L and $\overline{L}$ are Turing-recognizable $\Rightarrow$ $\exists$ M_1 that recognizes L and $\exists$ M_2 that recognizes $\overline{L}$. We use Turing machines M_1 and M_2 to build a decider M for L as follows:

M = On input w, where w is a string:

1. Run both M_1 and M_2 on input w in parallel.
2. If M_1 accepts, accept; if M_2 accepts, then reject.

Running M_1 and M_2 in parallel simply means that M has two tapes, one for simulating M_1 and one for simulating M_2.

Note that for any string w, either $w \in L$ or $w \in \overline{L}$, which means either M_1 or M_2 accepts w $\Rightarrow$ M either accepts or rejects any string. In fact, M accepts w $\iff$ $w \in L$ by construction $\Rightarrow$ M is a decider for L.

$\Rightarrow L$ is Turing decidable. (q.e.d.)

Corollary $\overline{L_{TM}}$ is not Turing-recognizable.

Proof We proved L_{TM} is Turing-recognizable. If $\overline{L_{TM}}$ were Turing-recognizable, then L_{TM} would be both Turing-recognizable and co-Turing-recognizable $\Rightarrow$ by the previous theorem, L_{TM} would be Turing-decidable $\Rightarrow \Leftarrow$ as we proved the contrary $\Rightarrow \overline{L_{TM}}$ is not Turing-recognizable, and we have constructed our example of a non Turing-recognizable language.

(q.e.d.)