Multiplying two integers using a function $\text{COPY}$

$$0^m 10^n \quad \text{COPY} \quad B^{m-1} 0^m$$

Idea: Use $\text{COPY}$ to copy $0^n$ $m$ times

$$0^m 10^n \quad \text{COPY} \quad 0101$$

$$B \quad 0^m \quad 10^n \quad 10^n \quad \text{COPY} \quad BB \quad 0^{m-2} \quad 10^n \quad 0^{2n}$$
$q_0: 0^n 10^i i^* 0^n 10^i$

9. To make an additional sequence \(0^n\)

\[
O^i q, 0^n 10^i \xrightarrow{O^j} 0^i q, 0^n 10^{i+n}
\]

9. Change a 0 \(O^j \) to B for countdown and enter state \(q_2\)

\[q_2: \text{Change a 0 in } 0^n \text{ to 2 and copy 0 to right}\]

9. Move left until 2, change to \(q_2\) and repeat until \(O^j 2^n 10^{i+n}\)

\[q_4: O^{j-1} 1 2^n 10^{i+n} \xrightarrow{O^{j-1}} 0^n 10^{i+n}\]
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9_1</td>
<td>9_2, 2, R</td>
<td>9_4, 1, L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9_2</td>
<td>9_2, 0, R</td>
<td>9_2, 1, R</td>
<td>9_3, 0, L</td>
<td></td>
</tr>
<tr>
<td>9_3</td>
<td>9_3, 0, L</td>
<td>9_3, 1, L</td>
<td>9_1, 2, R</td>
<td></td>
</tr>
<tr>
<td>9_4</td>
<td></td>
<td>9_5, 1, R</td>
<td>9_4, 0, L</td>
<td></td>
</tr>
</tbody>
</table>

\[
\delta (9_0, 0) = (9_1, 8, R)
\]
\[
\delta (9_6, 0) = (9_6, 0, R)
\]
\[
\delta (9_6, 1) = (9_1, 1, R)
\]
How about non-det TM?
Are they more powerful?

$\delta(q_i, a)$ contains
\[
\begin{cases} 
(q_i, b, R) \\
(q_{i_2}, c, L) \\
(1) 
\end{cases}
\]

How do we simulate using multi-tape det. TM?

$S, s_2, s_3, s_4, \ldots, s_m$

$S_i \in \{1, \ldots, 10\}$
Given a TM \( M \),
\[ L(M) = \{ w \mid M \text{ accepts } w \} \]

What happens if \( M \) doesn't accept \( w \) ?

- It may reach an undefined state
- It goes on forever
  - ID resetting
  - Head keeps moving right
- The class of languages accepted by a TM is called recursively enumerable (i.e.)

- The class of languages to which we can design a TM that always stops recursive languages
An alternate model of TM

Special tape: output tape

\[ \text{write a symbol and move right} \]

Output tape (read only tape input)
A *generator* TM writes out all the strings of a given language on the output tape.

A generator for a language \( L \) is guaranteed to write out \( w \in L \) on the output tape "eventually." A string \( w \notin L \) will never be written.

**Claim:** There is a generator \( G(L) \) for every r.e. language \( L \)

*Canonical ordering:* order according to lengths of strings

\[ S_1, S_2, S_3, \ldots \]

**Given a TM** \( M \) for \( L \):

- Run \( M \) on \( S_i \); if it accepts, write out \( S_i \) on output tape.

**Modification:** Run \( M \) for \( j \) steps on \( S_i \)
Generate all pairs \((i,j)\)

Run \(M\) for \(j\) steps on string \(s_i\)

if accepted write it out on output tape

Note that any string \(w + L\) will correspond to some \((i,j)\)
Claim: Given a generator $G(L)$ for a language $L$, then $L$ is r.e. (i.e. accepted by some TM $M$)

Proof. Run the generator and accept $w$ if $w$ is written on the output tape.

Generator for recursive languages.

The strings of a recursive language $L$ can be enumerated in the canonical ordering.

L, Canonical ordering

$w_i, w_{i_2}, w_{i_3}, \ldots w_{i_k}

i_1 < i_2 < i_3 < \ldots$

Show equivalence between TM that always stops (recursive) and generates that print strings in canonical order