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

$$0^m 10^n \rightarrow B^{m+n} 0^m$$

Idea: Use $\text{COPY} \cdot \text{COPY}$

$$0^n \text{ } m \text{ } \text{times}$$

$$0^m 10^n \rightarrow 010^1 0^m \text{ } \overbrace{\text{COPY}} \rightarrow B^0 10^n 10^n$$

$$B 0^{m-1} \rightarrow 0^n 10^n \text{ } \overbrace{\text{COPY}} \rightarrow BB 0^{m-2} 10^n 10^{2n}$$
\[ q_0 0^n 10^i \rightarrow 30^{m-1} i 9, 0^n 10^i \]

\( q_1 \) to make an additional sequence \( 0^n \)

\[ O^d 19, 0^n 10^i \rightarrow O^d 19, 0^n 10^{i+n} \]

\( q_2 \) : Change a 0 \( \rightarrow \) 0 \( \text{for countdown} \) and enter state \( q_2 \)

\( q_3 \) : Change a 0 in \( 0^n \) to 2 and copy 0 to right

\( q_4 \) : Move left until 2; change to \( q_2 \) and repeat

until \( O^d 12^n 10^{i+n} \)

\( q_5 \) : \( O^d 12^n 10^{i+n} \rightarrow O^d 10^n 10^n \)
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9₁</td>
<td>9₂,2,R</td>
<td>9₃,1,L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9₂</td>
<td>9₂,0,R</td>
<td>9₂,1,R</td>
<td>9₃,0,L</td>
<td></td>
</tr>
<tr>
<td>9₃</td>
<td>9₃,0,L</td>
<td>9₃,1,L</td>
<td>9₁,2,R</td>
<td></td>
</tr>
<tr>
<td>9₄</td>
<td></td>
<td>9₅,1,R</td>
<td>9₄,0,L</td>
<td></td>
</tr>
</tbody>
</table>

\[\delta(9₀,0) = (9₁,8,R)\]
\[\delta(9₆,0) = (9₆,0,R)\]
\[\delta(9₆,1) = (9₁,1,R)\]
How about non-deterministic TM?
Are there more powerful
\[ \delta(q_i, a) \text{ contains } \begin{cases} (q_i', b, R) \\ (q_{i_2}, c, L) \end{cases} \]

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

\[ \beta_1, \beta_2, \beta_3, \ldots, \beta_m \quad \beta_i \in \{1, \ldots, 103\} \]
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 repeating
  - Head keeps moving right
- The class of languages accepted by a TM is called recursively enumerable (r.e.)

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

Special tape: output tape

```
( ) → write a symbol and move right
output tape
```

```
read only tape input
```
A generator $TM$ writer 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 $L$.

**Modification:** Run $M$ for $j$ steps on $S_i$. If it accepts, write out $S_i$ on output tape.
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 \(w \in L\) 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.

$L = \{\text{Canonical order } w_1, w_2, w_3, \ldots w_k\}$

$i_1 < i_2 < i_3 < \ldots$}

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