INSTRUCTIONS

- You have 10 minutes to complete this quiz.
- The exam is closed book, closed notes, closed computer, closed calculator.
- This redemption quiz is not worth any points; turn it in at lecture if you would like feedback.
- Mark your answers on the exam itself. We will not grade answers written on scratch paper.
- For multiple choice questions,
  - □ means mark all options that apply
  - ○ means mark a single choice

<table>
<thead>
<tr>
<th>Last name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First name</td>
<td></td>
</tr>
<tr>
<td>Student ID number</td>
<td></td>
</tr>
<tr>
<td>CalCentral email (<a href="mailto:_@berkeley.edu">_@berkeley.edu</a>)</td>
<td></td>
</tr>
<tr>
<td>Teaching Assistant</td>
<td>Alex Stennet, Angela Kwon, Ashley Chien, Joyce Luong, Karthik Bharathala, Kavi Gupta, Kelly Chen, Michael Gibbes, Michelle Hwang, Mitas Ray, Rocky Duan, Samantha Wong</td>
</tr>
<tr>
<td>Name of the person to your left</td>
<td></td>
</tr>
<tr>
<td>Name of the person to your right</td>
<td></td>
</tr>
</tbody>
</table>

All the work on this exam is my own.
(please sign)
1. (5 points) Temmie Flakes (It’s just torn up pieces of construction paper.)

Implement enumerate_ways, which takes a tree t and an integer total and returns a list of the ways any sequence of consecutive nodes can sum to total. Below are the four ways included in enumerate_ways(t1, 7).

```python
def enumerate_ways(t, total):
    # Return a list of the ways that any sequence of consecutive nodes can sum to total.
    >>> t1 = tree(5, [tree(1, [tree(2, [tree(1)])]),
                      ...               tree(1, [tree(4, [tree(2, [tree(2)])])])],
                      ...               tree(3, [tree(2, [tree(2),
                      ...                        tree(3)])]),
                      ...               tree(3, [tree(1, [tree(3)])])])
    ...> enumerate_ways(t1, 7)
    [[5, 1, 1], [1, 4, 2], [3, 2, 2], [3, 1, 3]]
    >>> enumerate_ways(t1, 4)
    [[1, 2, 1], [4], [2, 2], [2, 2], [3, 1], [1, 3]]
    >>> t2 = tree(2, [tree(-10, [tree(12)]),
                      ...               tree(1, [tree(1),
                      ...                        tree(-1, [tree(2)])])])
    ...> enumerate_ways(t2, 2)
    [[2], [-10, 12], [2, 1, -1], [1, 1], [1, -1, 2], [2]]
    >>> enumerate_ways(t2, 4)
    [[2, -10, 12], [2, 1, 1], [2, 1, -1, 2]]
    def paths(__________________________________):
        ways = _________________________________
        if _____________________________________:
            ______________________________________
            for b in branches(t):
                ways += __________________________________________________________________________
                ______________________________________________________________________________________
                ______________________________________________________________________________________
        return ways
        return _____________________________________________________________________
```