INSTRUCTIONS

- You have 10 minutes to complete this quiz.
- The exam is closed book, closed notes, closed computer, closed calculator.
- Mark your answers on the exam itself. We will not grade answers written on scratch paper.
- For multiple choice questions, fill in each option or choice completely.
  - □ 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</td>
<td>(<a href="mailto:_@berkeley.edu">_@berkeley.edu</a>)</td>
</tr>
<tr>
<td>Discussion Section</td>
<td></td>
</tr>
<tr>
<td></td>
<td>___ ___ ___</td>
</tr>
</tbody>
</table>

All the work on this exam is my own. (please sign)

0. Your thoughts?  What can we do to make your learning experience better?
1. Copy Machine

(a) Peter wants to print this week's discussion handouts for all the students in CS 61A. However, both printers are broken! The first printer only prints multiples of \( n \) pages, and the second printer only prints multiples of \( m \) pages. Help Peter figure out whether or not it’s possible to print exactly total number of handouts!

```python
def has_sum(total, n, m):
    """
    >>> has_sum(1, 3, 5)
    False
    >>> has_sum(5, 3, 5)  # 0 * 3 + 1 * 5 = 5
    True
    >>> has_sum(11, 3, 5)  # 2 * 3 + 1 * 5 = 11
    True
    """

    if total == 0:  # (total == n or total == m) works too except when total equals 0
        return True
    elif total < 0:  # (total < min(n1, n2)) works given alternate base case
        return False
    return has_sum(total - n, n, m) or has_sum(total - m, n, m)
```

(b) The next day, the printers break down even more! Each time they are used, the first printer prints a random \( x \) copies \( 50 \leq x \leq 60 \), and the second printer prints a random \( y \) copies \( 130 \leq y \leq 140 \). Peter also relaxes his expectations: he’s satisfied as long as there’s at least lower copies so there are enough for everyone, but no more than upper copies to prevent waste.

```python
def sum_range(lower, upper):
    """
    >>> sum_range(45, 60)  # Printer 1 prints within this range
    True
    >>> sum_range(40, 55)  # Printer 1 can print a number 56-60
    False
    >>> sum_range(170, 201)  # Printer 1 + 2 will print between 180 and 200 copies total
    True
    """

def copies(pmin, pmax):
    if lower <= pmin and pmax <= upper:
        return True
    elif upper < pmin:
        return False
    return copies(pmin + 50, pmax + 60) or copies(pmin + 130, pmax + 140)
```

return copies(0, 0)