This midterm exam went reasonably well for most of you. The median was 45/55, which works out to 81.2%. The complete histogram of grades looks like this:

All scores of 30 and above are passing. We have a mapping of scores to letter grades; come see us in office hours if you want to talk in more detail about how you’re doing.

**Problem 1: Simple Python expressions (10 points)**

```python
>>> ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
>>> (9 - 8 + 7 + 6 * 5 % 4) ** 3 * 2 + 1
2001
>>> 2 % 7 == 7 % 5 or ALPHABET[len(ALPHABET)] == "Z"
True
>>> ALPHABET[-9::-13] + ALPHABET[4:2:-1]
'REED'
>>> str(ALPHABET.find("U")) * 2
20
2020
```  

The number 2020 is an important election year.

**Problem 2: Program tracing (10 points)**

```python
>>> Mystery
1
```  

The function `mystery(x, y)` implements a recursive version of the digital-root problem (Chapter 2, exercise 5, page 75).
Problem 3: Simple Python programs (15 points)

```python
def updateWord(guess, word, secret):
    """
    Returns a new version of the string word after filling in the
    character positions in which the guess appears in the secret word.
    """
    result = ""
    for i in range(len(word)):
        ch = word[i]
        if secret[i] == guess:
            ch = guess
        result += ch
    return result
```

Problem 4: Using the Portable Graphics Library (20 points)

```python
# File: MicroTetris.java
N_ROWS = 15
N_COLS = 10
SQUARE_SIZE = 30
GWINDOW_WIDTH = N_COLS * SQUARE_SIZE
GWINDOW_HEIGHT = N_ROWS * SQUARE_SIZE
TIME_STEP = 300

import random
from pg1 import GWindow, GRect

def MicroTetris():
    def clickAction(e):
        sx = square.getX()
        if e.getX() < sx:
            square.move(-SQUARE_SIZE, 0)
        elif e.getX() > sx + SQUARE_SIZE:
            square.move(SQUARE_SIZE, 0)

    def step():
        sx = square.getX()
        if square.getY() + SQUARE_SIZE >= GWINDOW_HEIGHT:
            dropNewSquare()
        else:
            square.move(0, SQUARE_SIZE)

    def dropNewSquare():
        nonlocal square
        square = GRect(SQUARE_SIZE, SQUARE_SIZE)
        square.setFilled(True)
        square.setColor("Red")
        gw.add(square, random.randrange(N_COLS) * SQUARE_SIZE, 0)

        gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT)
        square = None
        dropNewSquare()
        gw.addEventListener("click", clickAction)
        timer = gw.setInterval(step, TIME_STEP)

    # Startup code
    if __name__ == "__main__":
        MicroTetris()
```