HW 3
Prof. Shahed Sharif

Due 9/27: §2.8/14, 19, 20, 23, 24, 25, and the following Python problems:

A. Write a program `brute_caesar` that inputs a ciphertext and outputs the decryptions for every possible key, one per line. Use the program to decrypt "JVTLIVKYVJYZGDVEK."

B. Write a program `brute_affine` that does the same thing, but with the affine cipher and the text "JDGJIQFIVEGCATIUAAFUGJ".

19 The string "BA" is repeated 4 times, each time a multiple of 2 apart. Therefore we expect the key length to be 2. The first slice is "BBBAB". From the letter distribution, we expect that B stands for the most common letter b, and so A stands for a. The second slice is "AAAAA", so A should stand for b and B for a in this slice. Putting the slices back together, the plaintext is "bbbbbbabbb".

24 For the first part, all of the a_i need to be invertible mod 26: we decrypt each "slice" by a different affine cipher, so for this to be doable, we need the encryption function to be invertible, and hence a_i to be invertible. For the plaintext attack, we need 2 characters per affine cipher, which means a plaintext of length 10. The easiest case would be to use "AAAAABBBBB". As a numerical vector, this is

$$[0,0,0,0,0,1,1,1,1,1].$$

This encrypts to

$$[b_1,b_2,b_3,b_4,b_5,a_1+b_1,a_2+b_2,a_3+b_3,a_4+b_4,a_5+b_5].$$

So the first five entries give us the b_i. If we subtract these from the next five entries, we obtain the a_i.

B. The following does the trick:

```
def affine(a,b,L):
    """Apply affine transformation mod 26 to entries of L."""
    return [(a*x+b)%26 for x in L]

def affine_code(a,b,P):
    """Encrypt user input with affine code C=aP+b."""
    return num_to_str(affine(a,b,(str_to_num(P))))

def brute_affine(P):
    """Brute force affine cipher."""
    for a in range(25):
```

```
        if gcd(a,26) == 1:
            for b in range(25):
                print(affine_code(a, b, P))
    return
```

The functions `num_to_str`, `str_to_num`, and `gcd` are from the Python Tutorial homework. Note that the decryption function for an affine cipher is another affine cipher; of course, the `a`, `b` values will be different. The plaintext is "THATEULERWASONECOOLCAT".