HOMEWORK 1

CHRIS POWELL

1. In the Python tutorial, do exercises $4 - 9$.

Exercise 4. Find the sum of all numbers between 1 an 40 which are *not* multiples of 3; that is, $1 + 2 + 4 + 5 + \cdots + 40$.

```
def special_sum(n):
    """Compute sum from 1 to n, not multiples of 3, using while."""
    sum, i = 0, 0
    while (i < n):
        i += 1
        if i % 3 != 0:
            sum = sum + i
    return(sum)
```

Proof. Let i_k be the value of `i` after k iterations. Assume $n \in \mathbb{N}_{>0}$. As the value of `i` is incremented by one at each iteration, we get the recurrance relation $i_{k+1} = i_k + 1$. This implies that for $k \in \mathbb{N}$, $n - i_{k+1} < n - i_k$. But since $n \in \mathbb{N}_{>0}$, we know that for k sufficiently large $n - i_k \leq 0$. Thus the program must eventually terminate. □

Exercise 5. Write a new version of `dumb_multiply` which uses a **for** loop.

```
def dumber_multiply(a, b):
    """Multiply positive integers a and b using for."""
    p = 0
    for i in range(1, b + 1):
        if b != 0:
            p, b = p + a, b - 1
    return p
```

Exercise 6. Write a program which computes $n!$.

```
def factorial(n):
    """Compute n factorial using for."""
    p = 1
    for i in range(1, n + 1):
        p = p * i
    return(p)
```

Date: 2019.25.

Proof. Since the program uses a **for** loop, we know the program must eventually terminate. Let p_n be the value of `p` after n iterations. We show by induction on n that $p_n = n!$. Since

$$p_1 = p_0 \cdot 1 = 1 \cdot 1 = 1 = 1!$$

the statement is true for $n = 1$. Assume the statement holds for $n \in \mathbb{N}_{>1}$. Then $p_n = p_{n-1} \cdot n = n!$. Thus

$$p_{n+1} = p_n \cdot (n+1) = n! \cdot (n+1) = (n+1)!.$$

□

Exercise 7. Write a program which computes the nth Fibonancci number. If you used a **while** loop for the factorial problem, use a **for** for this one, and vice versa.

```
def pisano(n):
    """Compute nth fibonacci number using while."""
    if n == 1:
        return 0
    a, b = 0, 1
    i = 0
    while i < n:
        i += 1
        a, b = b, a + b
    return(a)
```

Proof. By applying an argument similar to the one given in Exercise 4, we can show that the program must eventually terminate for each positive integer input `n`. Let i_n, a_n, and b_n be the values stored by `i`, `a` and `b`, respectively, after n iterations. At each iteration, we have that $i_n = i_{n-1} + 1$ and $a_n = b_{n-1}$ and $b_n = a_{n-1} + b_{n-1}$. This implies $a_n = a_{n-1} + a_{n-2}$. We prove by strong induction. By noting that $a_2 = a_1 + a_0 = 0 + 1 = 1$, we establish the base case. Suppose a_k gives the correct fibonacci number for all $k \leq n$, where $k, n \in \mathbb{N}_{>2}$. Then $a_{k+1} = a_k + a_{k-1}$ gives the $k+1$ fibonacci number. □

Exercise 8. Write a program which takes as input a positive integer n and outputs the sum of the nth and $2nth$ Fibonacci numbers.

```
def leonardo(n):
    """Compute the sum of the nth and 2nth fibonacci numbers."""
    return pisano(n) + pisano(2*n)
```

```
def sod(x):
    s = 0
    while x > 0:
        s = s + (x % 10)
        x = x // 10
    return s
```

Exercise 9. Determine what the above program does **without** a computer, as follows.

(a) The values of `x`, `s` vary throughout the program. Given that we evaluate `sod(54132)`, construct a table with the values of `x` and `s`, where each row corresponds to the values as the program iterates.

i	`s`	`x`
0	0	54132
1	2	5413
2	5	541
3	6	54
4	10	5
5	15	0

(b) Since there is a while, it is not obvious that the program terminates - that is, that the program does not go into an infinite loop. Prove that, for any positive integer input x, the program does eventually terminate.

Proof. The program terminates when the value stored by `x` is non-positive. Let x_n be the value of `x` after n iterations. Assume $x_0 > 0$. Write $x_0 = \sum_{i=0}^{k-1} d_i 10^i$, where each $d_i \in \{0, \ldots, 9\}$. Then for $n \in \mathbb{N}_{>0}$,

$$x_n = x_0 - \sum_{i=0}^{n-1} d_i 10^i.$$

Thus

$$x_k = x_0 - \sum_{i=0}^{k-1} d_i 10^i = x_0 - x_0 = 0.$$

□

(c) Let x_n, s_n be the values of `x`, `s` after n iterations of the loop. For example, $s_0 = 0$. Determine what x_n and s_n are. (Normally, you would prove the claim by induction, but you do not have to do this here.)

$$s_n = \sum_{i=0}^{n-1} d_i \quad \text{and} \quad x_n = 0$$

(d) Using the previous parts, state what the program does, and prove your claim.

For any positive integer input $x = \sum_{i=0}^{n-1} d_i 10^i$, where $d_i \in \{0, \ldots, 9\}$, the program returns the sum of the digits of x; that is, $\sum_{i=0}^{n-1} d_i$. Otherwise, the program returns 0.

(e) What does sod stand for?

Presumably, `sod` stands for "sum of digits".

2. A grasshopper sits at the 0 of a number line. It can jump either 5 units in either direction, or 3 units in either direction, or any combination thereof (like two jumps of length 5 left, then a jump of length 3 right). What is the set of numbers that the grasshopper can reach with a sequence of jumps?

The set of numbers the grasshopper can reach is $\mathbb{Z}$.

Claim. For every integer k, there exist integers x and y such that $k = 3x + 5y$.

Proof. Clearly, $(m, n) = (2, -1)$ is a solution to $1 = 3m + 5n$. Let $k \in \mathbb{Z}$. Then $k \cdot (3m + 5n) = k \cdot 1$. But $\mathbb{Z}$ is a commutative ring, so

$$k \cdot (3m + 5n) = k(3m) + k(5n) = 3(km) + 5(kn).$$

Since 1 is the multiplicative identity in $\mathbb{Z}$, $k \cdot 1 = k$. Also, since $\mathbb{Z}$ is closed under multiplication, we know $km = x, kn = y$ for some $x, y \in \mathbb{Z}$. Hence $k = 3x + 5y$, as claimed. □

Therefore, since the grasshopper can get to 1 on the number line by some combination of jumps (namely, jumping 3 units to the right twice, and 5 units to the left once), it can get to any position on the number line by jumping some multiple of that combination.

3. Silverman 1.2. Give a geometric proof, and also a proof by induction.

Try adding up the first few odd numbers and see if the numbers you get satisfy some sort of pattern. Once you find the pattern, express it as a formula. Give a geometric interpretation.

Claim. The *nth* square is the sum of first n odd numbers.

Proof. We prove by induction on n. Clearly, the claim holds for $n = 1$ since 1 is odd and $1^2 = 1$. Suppose there exists some $n \in \mathbb{N}_{>1}$ such that $n^2 = \sum_{k=1}^{n}(2k-1)$. Then

$$n^2 + 2n + 1 = \left(\sum_{k=1}^{n} 2k - 1\right) + 2n + 1 = \sum_{k=1}^{n+1}(2k - 1).$$

But $(n+1)^2 = n^2 + 2n + 1$, so

$$(n+1)^2 = \sum_{k=1}^{n+1}(2k - 1).$$

Hence the claim is true for all $n \in \mathbb{N}$. □

Proof. (*Geometric*) Let S_n be the number of objects in an array with n rows and n columns, where $n \in \mathbb{N}_{>0}$. Then $S_n = n^2$ and $S_{n+1} = (n+1)^2 = n^2 + 2n + 1$. So $S_{n+1} - S_n = 2n + 1$. □

4. Silverman 1.3.

The consecutive odd numbers 3, 5, and 7 are all primes. Are there infinitely many such "prime triplets"? That is, are there infinitely many prime numbers p such that $p + 2$ and $p + 4$ are also primes?

No. In fact, $p = 3$ is the unique prime integer for which $p + 2$ and $p + 4$ are also primes.

Proposition. 3, 5, 7 are the only consecutive odd prime integers.

Proof. Let $p \neq 3$ be an odd prime integer. Then $p \not\equiv 0 \mod 3$ since 3 is the only prime which divides 3. So either $p \equiv 1 \mod 3$ or $p \equiv 2 \mod 3$. But if $p \equiv 1 \mod 3$, then $p = 3k + 1$ for some $k \in \mathbb{Z}$ which implies

$$p + 2 = 3k + 1 + 2 = 3k + 3 = 3(k + 1).$$

On the other hand, if $p \equiv 2 \mod 3$, then $p = 3k + 2$ for some $k \in \mathbb{Z}$ which implies

$$p + 4 = 3k + 2 + 4 = 3k + 6 = 3(k + 2).$$

Therefore, either $3 \mid p + 2$ or $3 \mid p + 4$. The result follows. □

E-mail address: powel054@cougars.csusm.edu