Name:

Instructions

- For maximum credit you must write your name on *all pages* of the exam.
- You have the full class period to complete the exam. Extra time will only be given for students with learning disablilities officially recognized by the University.
- Laptops, netbooks, smartphones, video game consoles, etc. may not be used during the exam. Turn them off/put them to sleep and keep them out of view. Simple calculators are allowed, but should mostly be unecessary. Whether or not a device qualifies as a simple calculator is entirely up to the instructor.
- No communicating with any other student or looking at any other student's test while taking the exam.
- You are free to leave once you turn in your exam.

Code tracing

Use the following piece of code to answer question 1

```
def sequence(n):
    while n != 1:
        print(n)
        if n % 2 == 0:
            n = n/2
        else:
            n = n * 3 + 1
```

1. What is printed to the screen if `sequence(12)` is called? (10 points)

Name:

Use the following piece of code to answer question 2 **and** 3

```
def bits(n):
    s = ''

    while n != 0:
        if n % 2 == 0:
            s = '0' + s
        else:
            s = '1' + s

        n = int(n/2)

    return s
```

2. What is the return value of `bits(8)`? (10 points)

3. What is the return value of `bits(11)`? (10 points)

Name:

Use the following peice of code for questions 4 **thru** 5

```
def f(n):
    if n == 0:
        return 0
    elif n % 2 == 0:
        return f(n) + n
    else:
        return f(n - 1) - n
```

4. There is a problem with the above function. What is it? Give one way to fix this problem (you may simply cross out the broken line and then write the corrected version of that line next to it or in the space below.) (10 points)

5. Show the stack diagram *of your fixed version* for the call `f(5)`. (10 points)

Name:

Code Writing

```
def saying_one_____________________:
    x = -1
    while ____________________:
        x = random.randint(1, 4)

    part = ''
    if x == 1:
        part = 'liver'
    elif x == 2:
        part = 'spleen'
    elif x == 3:
        part = 'duodenum'
    elif x == 4:
        part = 'femur'
    else:
        part = 'invalid'

    return prefix + ' my ' + part + "'s acting up"
```

6. Fill in the blanks in the above piece of code to make the assertion `assertEqual(saying_one('Aww jeez'), "Aww jeez my duodenum's acting up")` hold. (10 points)

```
def saying_two():
    direction = ''
    k = 0

    while ________________________________:
        if k % 2 == 0:
            direction = 'West'
        elif k < 8:
            direction = 'North'
        elif k % 7 == 0:
            direction = 'East'
        elif k % 3 == 0:
            direction = 'South'

        k = k + 1

    if k % 2 == ___ and k % 7 == ___:
        land = 'Peninsula'
    else:
        land = 'Island'

    return direction + 'most ' + land + ' is the Secret'
```

7. Fill in the blanks on the above piece of code to make the assertion `assertEqual(saying_two(), 'Eastmost Peninsula is the Secret')` hold. (10 points)

Name:

8. Implement a function `primes_between` which takes two integers x and y (such that $x < y$) and *prints to the screen* all prime numbers between x and y. (15 points)

9. Implement a function `digits` which takes an integer n and returns the number of digits in n. Your function should be able to handle negative integers. (15 points)