Math 121 Midterm

You are allowed 50 minutes for this midterm. Write answers clearly. All questions have equal weight. Remember that when writing code, the most important thing is correctness, but some attention will be paid to style. Simple, elegant, efficient and easy-to-understand code is best. There are four questions plus a bonus question to answer if you can.

Name _____

Question 1

Below, write what will appear in the terminal when the following code is run.

```
def foo(f):
    def bar(g):
        return f(f(g))
    return bar
f = foo(lambda x: x + 2)
print(f(2))
g = foo(print)
print(g(3))
```

```
6
3
None
None
```

Question 2

Write below code to define the function `everyOther`. This function takes as input a single list. The function then returns a list of every other element of the input list, starting with the second. You should return a new list. Do not modify the list that was given as input.

For example:
`everyOther([3, 7, 10, 8, 4])` returns `[7, 8]`.
`everyOther([9])` returns `[]`.
`everyOther([8, 7, 3, 5, 1, 4, 2, 6])` returns `[7, 5, 4, 6]`.

```
def everyOther(a):
    index = 1
    b = []
    while index < len(a):
        b.append(a[index])
        index += 2
    return b
```

Question 3

Write below code to define the function `count`. This function takes as input a list and a test function. (A test function takes one input and returns `True` or `False`.) It then returns the number of items of the list for which the test returns `True`. This function must be **recursive**.

For example:
If we define `atLeastTen` as follows…

```
def atLeastTen(n):
    return n >= 10
```

…then
`count([3, 17, 10, 8, 4], atLeastTen)` returns `2`.
`count([])` returns `0`.

```
def count(a, test):
    if len(a) == 0:
        return 0
    elif test(a[0]):
        return 1 + count(a[1:], test)
    else:
        return count(a[1:], test)
```

Question 4

Below is the code for the `Student` class we (roughly) defined in class. Write the code for a new class, `HungryStudent`, to represent a student that also eats food. This class should inherit from Student, but should differ in two ways. First, hungry students have some amount of money, which always starts at $100. Second, there is now an `eat` method. When `eat` is called, the student spends money on food, decreasing their money by $5 but increasing their energy by 3. If the student does not have the necessary $5, nothing (including the behavior described above) should happen.

```
class Student:
    def __init__(self, name):
        self.name = name
        self.energy = 10
        self.assignments = []
        self.knowledge = 0
    def getAssignment(self, size):
        self.assignments.append(size)
    def doAssignment(self):
        if self.assignments[0] <= energy:
            self.energy -= self.assignments[0]
            self.knowledge += self.assignments[0]
            self.assignments = self.assignments[1:]


class HungeryStudent(Student):
    def __init__(self, name):
        self.money = 100
        Student.__init__(self, name)
    def eat(self):
        if self.money >= 5:
            self.money -= 5
            self.energy += 3
```

Bonus

Write below code to define the function `maxPrimeFactor`. This function takes as input input a number and outputs the largest prime factor of that number.

For example:
1265 factors as 5 × 11 × 23, so `maxPrimeFactor(1265)` returns `23`.

```
def maxPrimeFactor(n):
    i = 2
    while i < n:
        if n % i == 0:
            n = n // i
        else:
            i += 1
    return i
```