CSCI2243-Assignment 4

Assigned September 26, due Sunday, October 1 at 11PM

This assignment is based on the last section of Chapter 3. You are asked to show that certain infinite sets are countable or uncountable. Except for the first two problems, 'showing' countability means providing a computer program that generates an infinite sequence that contains every element of the set exactly once.

Some direction for the computer problems: For each problem, you should have a single function with *no* arguments. The function should contain an inifinite loop headed by

```
while True:
```

It is ok, and often necessary, to have looping structures within the main `while` statement. Since these programs run forever, it is a good idea to put in a little thing that makes the program shut up after the first, say, 500 elements are printed, without being obliged to generate a keyboard interrupt. You can put

```
count=0
```

before the `while` statement, and

```
count+=1
if count>500:
        break
```

as the last statements in the `while` loop.

1. Problem 27 from Chapter 3. (The Hilbert Hotel.) Some notation you can use. Show the initial allocation of guests in the filled hotel as

   $$1 \quad 2 \quad 3 \quad \cdots$$

   This means guest 1 in room 1, guest 2 in room 2, etc. For part (a), show what the hotel room assignments look like after the arrival of a customer, whom we will denote a_1. For part (b), show the assignments after the arrival of a busload of customers

   $$b_1, b_2, b_3, \ldots.$$

For part (c), show the assignments after the arrival of customers

$$c_{ij}; i, j \in \mathbf{Z}^+,$$

where c_{ij} denotes the j^{th} passenger on the i^{th} bus. So the end result should show how the original customers, the new customer from part (a), the busload from part (b) and infinitely many busloads from part (c), are situated in the hotel, along with a few words explaining your scheme.

2. Problem 28. HINT: Remember, $\mathbf{Q}$ is countable, $\mathbf{R}$ is uncountable.

3. Problem 26. The trick here is weeding out the duplicates. If you wish, you may borrow the following snippet of code from a later chapter—a function that finds the greatest common divisor of two integers.

```
def gcd(m,n):
    while m!=0:
        (m,n)=(n%m,m)
    return n
```

Alternatively, you can maintain a list of the fractions printed so far, and check to see if each new positive rational number has already been printed.

4. Problem 25. This is harder. The output, when you run the program, should be a sequence of lines like

.
.
5
1,3,1
2,4
.
.

(although not, of course, in that order!) which, if continued indefinitely, would include all finite sequences of positive integers. The hint given in the textbook is not terribly helpful here, as it is an approach to proving that this set is countable, rather than how to write the program. In your writeup, explain your enumeration scheme. You will get partial credit for this problem if you describe a correct scheme but don't succeed in implementing it.