# ITERATORS, GENERATORS, AND STREAMS

COMPUTER SCIENCE MENTORS 61A

November 7 to November 11, 2016

## 1 Iterators

1. What is difference between an iterable and an iterator?

2. **Accumulator** Write an iterator class that takes in a list and calculates the sum of the list thus far.

```
>>> accu = Accumulator([1, 2, 3, 4, 5, 6])
>>> for a in accu:
...     print(a)
1
3
6
10
15
21
```

3. Is this an iterator or an iterable or both?

4. (Optional) Write `Accumulator` so it works if it takes in any iterable, not just a list

## 2 Generators

1. What does the following code block output?

```
def foo():
    a = 0
    if a < 10:
        print("Hello")
        yield a
        print("World")

for i in foo():
    print(i)
```

2. How can we modify `foo` so that `list(foo()) == [1, 2, 3, . . . , 10]`? (It's ok if there are extra `print`s)

3. Define `hailstone_sequence` a generator that yields the hailstone sequence. Remember, for the hailstone sequence, if n is even, we need to divide by two, otherwise, we will multiply by 3 and add by 1.

```
; Doctests:
>>> hs_gen = hailstone_sequence(10)
>>> hs_gen.__next__()
10
>>> next(hs_gen) #equivalent to previous
5
>>> for i in hs_gen:
>>> print(i)
16
8
4
2
1
```

4. (Optional) Define `tree_sequence`, a generator that iterates through a tree by first yielding the root value and then yield each branch.

```
>>> tree = Tree(1, [Tree(2, [Tree(5)]), Tree(3, [Tree(4)])])
>>> print(list(tree_sequence(tree)))
[1, 2, 5, 3, 4]
```

## 3 Streams

1. Whats the advantage of using a stream over a linked list?

2. Whats the maximum size of a stream?

3. Whats stored in first and rest? What are their types?

4. When is the next element actually calculated?

5. For each of the following lines of code, write what Scheme would output.

```
scm> (define x 1)

scm> (if 2 3 4)

scm> (delay (+ x 1))

scm> (define (foo x) (+ x 10))

scm> (define bar (cons-stream (foo 1) (cons-stream (foo 2)
   bar)))

scm> (car bar)

scm> (cdr bar)

scm> (define (foo x) (+ x 1))

scm> (cdr-stream bar)

scm> (define (foo x) (+ x 5))
```

```
scm> (car bar)

scm> (cdr-stream bar)
```

6. Write out `double_naturals`, which is a stream that evaluates to the sequence 1, 1, 2, 2, 3, 3, etc.

```
(define (double_naturals)
    (double_naturals_helper 1 0)
)

(define (double_naturals_helper first flag)









)
```

7. Write out `interleave`, which returns a stream that alternates between the values in stream1 and stream2. Assume that the streams are infinitely long.

```
(define (interleave stream1 stream2)









)
```