1. What does the following code block output?

```python
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 okay if the program prints along the way.)
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 multiply by 3 and add by 1.

```python
def hailstone_sequence(n):
    """
    >>> 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. Define `tree_sequence`, a generator that iterates through a tree by first yielding the root value and then yielding the values from each branch.

```python
def tree_sequence(t):
    """
    >>> t = Tree(1, [Tree(2, [Tree(5)]), Tree(3, [Tree(4)])])
    >>> print(list(tree_sequence(t)))
    [1, 2, 5, 3, 4]
    """
```
2 Streams

1. What’s the advantage of using a stream over a linked list?

2. What’s the maximum size of a stream?

3. What’s 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.

```scheme
scm> (define x 1)

scm> (if 2 3 4)

scm> (define p (delay (+ x 1)))

scm> p

scm> (force p)

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.

\[
\text{(define (double_naturals)}
\text{  (double_naturals_helper 1 0))}
\]

\[
\text{(define (double_naturals_helper first go-next))}
\]

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

\[
\text{(define (interleave stream1 stream2)}
\]

\[
\text{)}
\]
Challenge Question

8. *(Optional)* Write a generator that takes in a tree and yields each possible path from root to leaf, represented as a list of the values in that path. Use the object-oriented representation of trees in your solution.

```python
def all_paths(t):
    """
    >>> t = Tree(1, [Tree(2, [Tree(5)]), Tree(3, [Tree(4)])])
    >>> print(list(all_paths(t)))
       [[1, 2, 5], [1, 3, 4]]
    """
```