Answer the questions in the spaces provided on the question sheets. You should have plenty of room for the answers, but if you run out of room because you need to make a correction, you can use the back of the page (and indicate that you are doing so).

The first page of the quiz is a reference for list and table operations. Unless otherwise noted in a question, feel free to use any of the operations defined in the reference. Please do not use operations that are not in the reference.

Anonymous grading ID: ____________________________________________

Name: __________________________________________________________

(Please include your name if and only if you don’t remember your anonymous ID)

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python printing</td>
<td>20</td>
</tr>
<tr>
<td>Getting into shape</td>
<td>20</td>
</tr>
<tr>
<td>Climbing trees</td>
<td>20</td>
</tr>
<tr>
<td>Total:</td>
<td>60</td>
</tr>
</tbody>
</table>
Python printing  (20 points)
Examine each of the Python programs below. For each program, write what its output will be when run–i.e., what will be printed? For example, the output of this program:

```python
print(3)
print(4)
```
would be:

3
4

(a) 5 points Program 1
```
x = 1
print(x)
x = x + 1
print(x)
```

(b) 5 points Program 2
```
def print_and_return(x : int) -> int:
    print(x)
    return x + 1

y = print_and_return(3)
print(y)
```

(c) 5 points Program 3
```
lst = [1, 2, 3]
for n in lst:
    print(n + 1)
```

(d) 5 points Program 4
```
lst = [1, 2, 3]
p = 1
for n in lst:
    p = p * n
print(p)
```
Getting into shape  (20 points)

Imagine you’re working on a graphics library in Pyret. Part of the library is a datatype for shapes; right now, you only need to support rectangles and circles. Rectangles should have both width and height (which are numbers), as well as a color (which is a string). Circles should have a radius (number) and a color (string).

(a) Define a Pyret datatype for shapes.
(b) 10 points Write a **Pyret** function called `area` that takes in a shape (as you’ve defined them) and returns its area. A rectangle’s area is its width multiplied by its height; a circle’s area is 3.14 times the square of its radius. Your function does not need to include a docstring or tests.
Climbing trees  (20 points)

This question will work with the following version of our Pyret AncTree class, to which we have added a height field representing the height in inches of a person in the tree:

```pyret
data AncTree:
    | unknown
    | person(name :: String, height :: Number,
              mother :: AncTree, father :: AncTree)
end
```

(a) 10 points Suppose we have defined the following Pyret function:

```pyret
fun mystery-function(tree :: AncTree) -> List<String>:
    cases (AncTree) tree:
        | unknown => [list: ]
        | person(nm, h, mo, fa) =>
            L.append(mystery-function(mo),
                      L.append(mystery-function(fa), [list: nm]))
    end
end
```

What would the following function call return?

```pyret
mystery-function(
    person("A", 68,
        person("B", 70, unknown, unknown),
        person("C", 67, unknown, unknown)))
```
(b) 10 points Write a Pyret function any-taller-people that takes an AncTree and a Number and returns a Boolean. The function should return true if any person in the tree has a larger height than the given number. Your function does not need to include a docstring or tests.