

## SOLUTIONS

### Problem 1.

```
def reverse(a):
    if a == []:
        return []
    if len(a) == 1:
        return a
    else:
        return [a[-1]] + reverse(a[0:-1])
```

### Problem 2.

```
def make_tuples(a, b):
    if a == [] or b == []:
        return []
    else:
        return [(a[0], b[0])] + make_tuples(a[1:], b[1:])
```

### Problem 3.

```
def sum_leaves(t):
    if t == None:
        return 0
    if t._left == None and t._right == None:
        return t._value
    else:
        return sum_leaves(t._left) + sum_leaves(t._right)
```

### Problem 4.

- a) An invariant is *something that is true* about the state of a program *at some point* in the code during execution.
- b)  $y$  is even  
 $y \leq x + 1$

### Problem 5.

```

# add a node to the head of the list
def add(self, node):
    node._next = self._head
    self._head = node

def remove(self):
    if self._head == None:
        return None
    else:
        prev = self._head
        cur = prev._next
        if cur == None:
            self._head = None
            return prev
        else:
            while (cur._next != None):
                prev = cur
                cur = cur._next

            prev._next = None
            return cur

class Queue:
    def __init__(self):
        self._items = LinkedList()

    def enqueue(self, item):
        self._items.add(Node(item))

    def dequeue(self):
        n = self._items.remove()
        return n._value

```

**Note:** `dequeue ()` as shown above returns the value of the node. You may have written this initially:

```

def dequeue (self):
    return self._items.remove()

```

However, this returns an object of type `Node`, which is not what the user expected. Even though we are using linked lists as the underlying representation, the user is pushing and popping elements of a given type and `dequeue ()` must return the element as it was when pushed on to the queue.

**Problem 6.**

0	1	2	3	4	5	6
14	2	23	10	24	19	

**Problem 7.**

a)

AAA

AB

BA

b) 1011111 (represents CABA)

**Problem 8.**

Various solutions, but the bug is exposed when `wordlist` is an empty list.

value for wordlist	value for tail
<code>[]</code>	<code>'anything'</code>
<code>['this', 'is', 'it']</code>	<code>'is'</code>
<code>['a', 'field', 'of', 'hay']</code>	<code>'go'</code>
<code>['a', 'field', 'of', 'hay']</code>	<code>'as'</code>