5) [8 pts] For the following snippets of code indicate **the first error it produces and give the line number of the error.** If there is no error, state that. If there is more than one error, identify the first error that would be encountered. Assume each snippet of code is executed independently.

Possible Errors: (each may be used more than once)

- KeyError
- AssertError
- NameError
- TypeError
- IndentationError
- SyntaxError

a)

```
1 my_dict = {'a': 4, 'b': 5, 'd': 7}
2 my_dict['a'] = "red"
3 print my_dict['c']
```

a) Error: KeyError

Line number: 3

b)

```
1 for i in range(19):
2     if i % 2 == 0:
3     print 'even'
4     else:
5     print 'odd'
6 assert(i % 2 == 0)
```

b) Error: IndentationError

Line number: 3

c)

```
1 lst = [3, 5, 2]
2 print "length" + len(lst)
3 assert(len(lst) == 3)
```

c) Error: TypeError

Line number: 2

d)

```
1 my_list = [1, 2, 3]
2 if 4 not in my_list:
3     the_list.append(4)
4 assert(4 in my_list)
```

d) Error: NameError

Line number: 3

2)[3 pts] Write code that would produce a "TypeError: 'list' object not callable" error

```
lst = [1,2,3]
lst(4)
```

4) [12 pts] Write at least two assert statements for each function below that you might use to test that the following functions are correctly implemented. If you think that the functions are incorrectly implemented, include assertions that will expose errors in the function.

```
def max(lst):
    """ Takes a list of integers and returns the largest value
    in the list. We consider the maximum value of an empty list
    to be None. """
    max = 0
    for x in lst:
        if x > max:
            max = x
    return max
```

```
assert max([-3, -5, -7]) == -3
assert max([]) == None
```

```
def avg(lst):
    """ Takes a list of integers and returns a float which is the
    average of the list. We consider the average value of an
    empty list to be None. """
    total = 0
    num_values = 0
    for x in lst:
        total += x
        num_values += 1
    return total / num_values
```

```
assert avg([]) == None
assert avg( [7, 8]) == 7.5
```

1) [4 pts] Give three advantages of using functions. Your reasons should be as different as possible, and no longer than a sentence each (a phrase each is fine).

- **Documentation: Gives a comprehensible name to a computation.**
- **Abstraction: Clients can depend on a short specification, and ignore details of the implementation.**
- **Re-use of code: In this program or programs you might write in the future.**
- **Aids Testing and debugging: enables divide and conquer.**

1) [5 pts] You receive the following error messages after running your code:

```
Traceback (most recent call last):
  File "social_network.py", line 338, in <module>
    do_stuff(friends_list)
  File "social_network.py", line 200, in do_stuff
    result = recommend_by_influence(friendlist)
  File "social_network.py", line 107, in recommend_by_influence
    output = read_result()
NameError: global name 'read_result' is not defined
```

a) List the names of the stack frames that existed at the point the error was discovered.

`Global, do_stuff, recommend_by_influence`

b) What is the most recent stack frame (e.g. the last function we successfully called)?

`recommend_by_influence`

c) Describe how you would go about trying to find the cause of and fix this error.

Most likely this is due to a misspelling of the function name referred to as "read_result()" on line 107 of social_network.py. So a good start would be to search to see if there is a similarly named function in the file social_network.py. If that fails, maybe this function is defined in another namespace like we did before with Random or nx, requiring the function name to be prefaced with that module name.