1. The code below does not have any errors. If you run this code in SageMath, what will the output be?

```python
results = [4]
for i in srange(0, 8):
    value = results[i]
    newvalue = 2*value - 3
    results.append(newvalue)

print results
```

**Solution:** This code iterates the function $f(x) = 2x - 3$ eight times, starting with an initial value of 4. The output will be the list of numbers [4, 5, 7, 11, 19, 35, 67, 131, 259].
2. You need to write code that will iterate the function \( f(x) = \frac{x^2}{10} \). It should start with an initial value of 3, iterate the function 20 times, and plot the resulting list of numbers. Choose a correct sequence of lines of code from the choices below to accomplish this.

**First line:**
A. values = []
B. values = [3]
C. value = 3
D. values = srange(0, 20)

**Second line:**
A. for n in values:
B. for n in count:
C. for n in (0, 20, 1):
D. for n in srange(20):

**Third line:**
A. a = values(-1)
B. a = values[n]
C. value = value^2 / 10
D. a = values[-1]

**Fourth line:**
A. values.append(a^2 / 10)
B. values.append(f(a))
C. results = [value]
D. results.append(value)

**Fifth line:**
A. list_plot(values)
B. list_plot(results)
C. list_plot(values)
D. plot(x^2 / 10, (x, 0, 20), ymin=3)

**Solution:** Valid sequences of lines:
- B, D, B, A, C
- B, D, D, A, C
3. The following script creates a list of values of the function \( f(x) = 2x^2 \), as \( x \) ranges from 1 to 10. In other words, if you run the code below, the output is [2, 8, 18, 32, 50, 72, 98, 128, 162, 200].

\[
\text{results} = [] \\
f(x) = 2*x^2 \\
\text{for } n \text{ in srange(1, 11)}: \\
\quad \text{newvalue} = f(n) \\
\quad \text{results.append(newvalue)} \\
\text{results}
\]

You need to modify this code to turn the script into a function, called \texttt{funclist}, which will take as inputs (arguments) the function \( f \) and the maximum value of \( x \) for the loop. For example, after defining this function, you could enter

\[
h(x) = x^3 \\
\text{funclist}(h, 5)
\]

The result should be the list \([1, 8, 27, 64, 125]\).

Exactly four modifications (other than indenting) are needed to make this change. Mark up those four changes, with the proper indentation, on the original script above.

**Solution:**

```python
def funclist(f, xmax):  # You could use any name in place of xmax 
    results = [] 
    \( f(x) = 2*x^2 \)  # Get rid of this line entirely 
    \text{for } n \text{ in srange(1, xmax + 1)}: 
    \quad \text{newvalue} = f(n) 
    \quad \text{results.append(newvalue)} 
    \text{return results}
```

```
4. The goal of the code below is to define a function, called `deriv_at`, that takes as input a symbolic expression $f$ (such as $x^3 + 5x$) and a number $a$. The function should return $f'(a)$. So, for example, if the function were called as `deriv_at(x^2, 5)`, the result should be 10. And if it were called as `7*deriv_at(x^2, 3)`, the result should be 42. However, the code below has five errors. Find all five of them, and explain how you would correct each one.

```python
var(x)
def deriv_at(f, a):
    fprime = diff(f)
    fprime.subs(x=a)
    print result
```

**Solution:**

```python
var("x")
def deriv_at(f, a):
    fprime = diff(f, x)
    result = fprime.subs(x=a)
    return result
```
5. The goal of the following code is to perform Euler’s method on the differential equation

\[ R' = \frac{R^2}{1 + R^2} - 0.2R \]

starting with an initial state of \( R = 2 \), with a time step of \( \Delta t = 0.1 \), and running for 1000 steps (so until \( t = 100 \)). It should then plot the results, with the correct times on the horizontal axis. However, the code below has six errors. Find all six of them, and explain how you would correct each one.

```python
Rprime(R) = R^2/1 + R^2 - 0.2*R
statelist = [2]
for i in srange(1000):
    current_state = statelist[i]
    change_vector = f(current_state)
    next_state = current_state + change_vector
    statelist.append(next_state)
timelist = srange(0, 100, 0.1)
list_plot((timelist, statelist), plotjoined=True)
```

Solution:

```python
Rprime(R) = R^2/(1 + R^2) - 0.2*R
statelist = [2]
for i in srange(1000):
    current_state = statelist[i]
    change_vector = Rprime(current_state)
    next_state = current_state + 0.1*change_vector
    statelist.append(next_state)
timelist = srange(0, 100, 0.1)
list_plot((timelist, statelist), plotjoined=True)
```