\[ n = 5 \]
\[ c = \text{range}(n) \]
\[ c = [x \text{ for } x \text{ in } \text{range}(n)] \]
\[ [0, 1, 2, 3, 4] \]
\[ c = [2 \times x \text{ for } x \text{ in } \text{range}(n)] \]
for each item in my list
value
\[ [0, 2, 4, 6, 8] \]
do something
\[ n = 5 \]
\[ c = [1 \text{ for } x \text{ in } \text{range}(n)] \]
\[ [0, 1, 2, 3, 4] \]
\[ c = [1 \text{ for } x \text{ in } \text{range}(n) \text{ if } x \% 2 == 0] \]
\[ [0, 1, 1, 1, 1, 1] \]
\[ \text{sum}(c) \]
\[ c = [1 \text{ for } x \text{ in } \text{range}(n) \text{ if } x \% 2 == 0] \]
equivalent code
\[ c = [] \text{ for } x \text{ in } \text{range}(n) \text{ create empty list}\]
if div by 2
for each item in old list
\[ c = [x \text{ for } x \text{ in } \text{range}(n) \text{ if } x > 2] \]
\[ \text{value is } [3, 4] \]
if meets this condition
\[ 
\]
**PROBLEM 1: (Smith [Corona—Wesson] (32 points))**

**Part A (22 points)**

Each of the variables below has a type and a value. The type is one of: list, boolean, int, string, float. For example, consider the assignment to variable `x` below:

\[ x = \text{len}([5,3,1]) \]

The type and value are shown in the first row of the table below. Fill in the other type and value entries based on the variable/expression in the first column.

<table>
<thead>
<tr>
<th>variable/expression</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = \text{len}([5,3,1])</code></td>
<td>int or integer</td>
<td>3</td>
</tr>
<tr>
<td><code>a = 27/15</code></td>
<td>int</td>
<td>1</td>
</tr>
<tr>
<td><code>h = 32 \% 11</code></td>
<td>int</td>
<td>10</td>
</tr>
<tr>
<td><code>e = \text{sum}([\text{range}(6)])</code></td>
<td>int</td>
<td>15</td>
</tr>
<tr>
<td><code>d = \text{len}(&quot;dog&quot;) &lt; \text{len}(&quot;catsup&quot;)</code></td>
<td>bool</td>
<td>true</td>
</tr>
<tr>
<td><code>c = &quot;1,234,567&quot;.\text{split}(&quot;,&quot;)</code></td>
<td>list</td>
<td>[&quot;1&quot;, &quot;234&quot;, &quot;567&quot;]</td>
</tr>
<tr>
<td><code>i = \text{&quot;platform&quot;}[-2]</code></td>
<td>String</td>
<td>&quot;r&quot;</td>
</tr>
<tr>
<td><code>g = \text{&quot;beat&quot;} + \text{&quot;nick&quot;}</code></td>
<td>String</td>
<td>&quot;beatnicking&quot;</td>
</tr>
<tr>
<td><code>b = \text{&quot;snapdragon&quot;}[3:8]</code></td>
<td>String</td>
<td>&quot;drag&quot;</td>
</tr>
<tr>
<td><code>f = 0.23*10</code></td>
<td>Float</td>
<td>2.3</td>
</tr>
<tr>
<td><code>k = [8,7,6,7,6,5,4,3,2,1][5:7]</code></td>
<td>list</td>
<td>[5, 4]</td>
</tr>
<tr>
<td><code>j = 2**4</code></td>
<td>int</td>
<td>16</td>
</tr>
</tbody>
</table>

\[ \text{truncates down to next integer} \]

-2, -1, 0, 1, 2

\[ \Rightarrow C \]

0, 1, 2

\[ \leftarrow \text{check this: } -2/3 \text{ is } -1 \]

I checked this: \(-2/3 \text{ is } -1\)
Part B (4 points)
The Rohrer's Index is an alternative to body mass index (BMI) as an anthropometric statistic. It is calculated using the formula below where weight is in pounds and height is in inches. Write the function rohrer whose header is provided below the formula. For example, rohrer(165, 73) is \( \frac{165 \times 2768}{73^3} = 1.174 \)

\[
\frac{\text{weight} \times 2768}{\text{height}^3}
\]

```python
def rohrer(weight, height):
    return float(value for rohrer's index given
    int parameters weight and height

val1 = 1.0 * weight
val2 = val1 * 2768
```

(continued)
Part C (6 points)

Heron's Formula for the area of a triangle with three sides $a, b, c$ can be calculated in two steps: the first calculates the semi-perimeter ($s$ below) and the second calculates the area ($A$ below):

$$s = \frac{a + b + c}{2}$$

$$A = \sqrt{(s-a) \times (s-b) \times (s-c) \times s}$$

For example, if the sides of a triangle are 3, 4, 5, then $s = (3 + 4 + 5)/2 = 6$ and the area is

$$\sqrt{(6-3)(6-4)(6-5)(6)} = \sqrt{3 \times 2 \times 1 \times 6} = \sqrt{36} = 6$$

Complete the function `triangle_area` that returns the area of a triangle with sides whose lengths are given by parameters $a$, $b$, and $c$ (To compute a square root use the function `math.sqrt` or raise a number to the 0.5 power). For example, `triangle_area(3, 4, 5)` should evaluate to 6.0.

```python
def triangle_area(a, b, c):
    s = (a + b + c) / 2.0
    area = math.sqrt((s-a) * (s-b) * (s-c) * s)
    return area
```

4
**PROBLEM 2 : (Accumulated Wisdom (18 points))**

**Part A (5 points)**

Write the function `divisor_sum` that returns the sum of the proper divisors of a number. For example:

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
<th>reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>divisor_sum(12)</code></td>
<td>16</td>
<td>(1 + 2 + 4 + 6 = 13)</td>
</tr>
<tr>
<td><code>divisor_sum(284)</code></td>
<td>220</td>
<td>(1 + 2 + 4 + 71 + 142 = 220)</td>
</tr>
<tr>
<td><code>divisor_sum(28)</code></td>
<td>28</td>
<td>(1 + 2 + 4 + 7 + 14 = 28)</td>
</tr>
</tbody>
</table>

```python
def divisor_sum(n):
    return int(sum of proper divisor of n, n is an int > 0)
    sum = 0
    for k in range(n):
        if n % k == 0:
            sum = sum + k
    return sum
```

*alt.*

```python
return sum([x for x in range(n) if n % x == 0])
```
Part B (5 points)

A pair of numbers is amicable if the sum of the proper divisors of one number is equal to the other, and vice versa. For example, the pair (220, 284) is amicable because the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, and 110; the proper divisors of 284 are 1, 2, 4, 71, 142; and we have

\[1 + 2 + 4 + 71 + 142 = 220\]
\[1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284\]

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_amicable(220, 284)</td>
<td>True</td>
</tr>
<tr>
<td>is_amicable(10, 35)</td>
<td>False</td>
</tr>
<tr>
<td>is_amicable(1184, 1210)</td>
<td>True</td>
</tr>
<tr>
<td>is_amicable(2214, 2688)</td>
<td>False</td>
</tr>
</tbody>
</table>

Write is_amicable below. You should call divisor_sum, assume it works as specified.

```python
def is_amicable(x, y):
    
    returns True if and only if int parameters x,y are amicable
    
    if divisor_sum(x) == y and divisor_sum(y) == x:
        return True
    else:
        return False
```

```
return \begin{align*}
\text{div-sum}(x) &= y \quad \text{and} \quad \\
\text{div-sum}(y) &= x \\
\end{align*}
```
Part C (8 points)

Cities or towns are specified by a three-element list such as ["San Jose", 37.2406, -121.7457] where the first element is a string and the second two elements are float values specifying the latitude and longitude of the city.

Write the function nearby that has three parameters: the first parameter city is one three-element list specifying a city, the second parameter clist is a list of three-element lists specifying a list of cities, and the third parameter apart is a float representing a distance in miles. The function should return a list of strings: the names of the cities in clist that are no more than apart miles from city. For example, the call below would return the list ["Palo Alto", "Big Sur"], the only cities in clist within 150 miles of San Jose.

```
nearby(["San Jose", 37.2406, -121.7457],
       [["Palo Alto", 37.2833, -121.9179],
        ["Princeton", 40.3436, -74.694],
        ["Big Sur", 35.9348, -121.46894],
        ["Los Angeles", 34.0522, -118.2428]],
       150)
```

In writing nearby you must call the function distance shown below that takes two coordinate pairs of latitudes and longitudes and correctly returns the distance in miles between the coordinates.

```python
def distance(la1,lo1, la2,lo2):
    ""
    return distance in miles between (la1,lo1) and (la2,lo2), latitude and longitude pairs
    ""
    x = 69.1*(la1-la2)
    y = 53.0*(lo1-lo2)
    return math.sqrt(x*x + y*y)

def nearby(city, clist, apart):
    cities = []
    for elt in clist:
        if distance(elt[1],elt[2],city[1],city[2]) <= apart:
            cities.append(elt[0])
    return cities
```

```
return [elt[0] for elt in clist if elt in cities]
```
PROBLEM 3:  (Big Ugly Gigantic Spiders (16 points))

Part A (4 points)
The function censored below is supposed to return "censored" if the string parameter headline contains any of the words in parameter bad, a list of strings, and return "clean" if headline does not contain any words in bad.

The table illustrates what censored is supposed to return.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>censored([&quot;red&quot;,&quot;blue&quot;,&quot;green&quot;], &quot;red tide awesome&quot;)</td>
<td>&quot;censored&quot;</td>
</tr>
<tr>
<td>censored([&quot;red&quot;,&quot;blue&quot;,&quot;green&quot;], &quot;cardinal rules&quot;)</td>
<td>&quot;clean&quot;</td>
</tr>
<tr>
<td>censored([&quot;big&quot;,&quot;bad&quot;,&quot;trouble&quot;], &quot;bad news is trouble&quot;)</td>
<td>&quot;censored&quot;</td>
</tr>
<tr>
<td>censored([&quot;big&quot;,&quot;bad&quot;,&quot;trouble&quot;], &quot;bigfoot badboy troublesome&quot;)</td>
<td>&quot;clean&quot;</td>
</tr>
</tbody>
</table>

The implementation below is not correct. However it passes three of the four tests/examples shown above. Explain which one test the implementation below fails and why it fails.

```python
def censored(bad,headline):
    words = headline.split()
    for b in bad:
        if b in words:
            return "censored"
    else:
        return "clean"
```

Part B (4 points)
Write a correct version of censored below - this should work for all parameters, not just those shown in the table above. You can modify the code above, or copy/change it below.