**Part (b)**  [2 marks] One instructor’s solution to Part (a) had a runtime that is best described as being quadratic in the length of the `words_to_phonemes` Python dict. If it took 1 second for the `build_rhyming_dict` function to run for a `words_to_phonemes` dict containing 1000 words, roughly how long would you expect the `build_rhyming_dict` function to take when the `words_to_phonemes` dict was doubled in length to a size of 2000 words? Justify your response.

**Question 9.**  [8 marks]

Consider the following Python function. The docstring has been shortend to save space.

```python
def bark_like_a_dog(L):
    """ (list of object) -> NoneType
    """
    for item in L:
        print('Woof!')
```

Each of the following sets of Python code operate on a list named L. You may assume that L refers to a list of objects and that `len(L)` is n. For each set of Python code, write a formula that expresses approximately how many times the word Woof! is printed. The formula may depend on n. In addition, circle whether the dependence on n is constant, linear, quadratic or something else.

<table>
<thead>
<tr>
<th>Python code</th>
<th>How many times is Woof! printed? (approximately)</th>
<th>Dependence on n (circle one)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bark_like_a_dog(L)</code></td>
<td>$n$</td>
<td>constant</td>
</tr>
</tbody>
</table>
| `i = 0` while `i < len(L)`:
  `bark_like_a_dog(L[i:i+1])`
  `i = i + 1`                        | $n \times 1 = n$                              | quadratic                    |
| `i = 0` while `not(i < len(L))`:
  `bark_like_a_dog(L)`
  `i = i + 1`                        | 0                                               | quadratic                    |