Chapter 6  Answers to some of questions

True/False Questions
1) False (recursive functions call themselves, there are quite useful ones)
2) True (page 212, Summary)
3) False (recursive functions must have at least one base case, i.e. it might have 1 or 2 or 3..)
6) True

<table>
<thead>
<tr>
<th>1st letter</th>
<th>2nd letter</th>
<th>3rd letter</th>
<th>...</th>
<th>(n-1)th letter</th>
<th>nth letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>n choices</td>
<td>(n-1) choices</td>
<td>(n-2) choices</td>
<td>n-((n-1)-1) = 2 choices</td>
<td>n-(n-1) = 1 choice</td>
<td></td>
</tr>
</tbody>
</table>

(“choices” stands for “letters to choose from”)
i.e. we get $n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1 = n!$

Multiple Choice questions
1) d)
4) d) (because 4! = 1*2*3*4 = 24)
5) c) (n iterations)
6) a) (the power is divided by two which gives $log n$ recursive calls)

Short-Answer questions
1) Yes. A proper recursive function must have at least one base case, which will be checked by a decision structure (control structure, like if ...)

Trace `recPower(4,8)` and figure out exactly how many multiplications it does.

```
recPower(4,8)
  factor = recPower(4,4)
  return factor * factor
  256

factor = recPower(4,2)
  return factor * factor
  16

factor = recPower(4,1)
  return factor * factor
  4

factor = recPower(4,0)
  return factor * factor * 4
  1

1*1 * 4 = 4
```
1) What list is returned by `anagrams("word")`?

**Preliminary analysis:**
First of all, we expect $4! = 24$ of them ($\frac{4!}{4\cdot3\cdot2\cdot1}$ - number of choices)

Secondly, we can simply run the program and see the result, but what's most important is to understand how that result was achieved!

**Pictorial representation of `anagram("word")` call:**

```
<table>
<thead>
<tr>
<th>anagrams(&quot;word&quot;)</th>
<th>anagrams(&quot;ord&quot;)</th>
<th>anagrams(&quot;rd&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[word, owrd, orwd, ordw, word, rwd, rowd, rodw, wrdo, rwdo, rdwo, rdw, wodr, ... wdo, ... , dro]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Result:** `['word', 'owrd', 'orwd', 'ordw', 'wrd', 'rowd', 'rodw', 'wrdo', 'rwdo', 'rdwo', 'rdw', 'wodr', 'owdr', 'odwr', 'odrw', 'wdr', 'dorw', 'dowr', 'dorw', 'wdro', 'dwro', 'drwo', 'drow']`

If you add `print(ans)` right before `return ans`, this is what you will see:

```
['d']
['rd', 'dr']
['ord', 'rod', 'rdo', 'odr', 'dor', 'dro']
['word', 'owrd', 'orwd', 'ordw', 'wrd', 'rowd', 'rodw', 'wrdo', 'rwdo', 'rdwo', 'rdw', 'wodr', 'owdr', 'odwr', 'odrw', 'wdr', 'dorw', 'dowr', 'dorw', 'wdro', 'dwro', 'drwo', 'drow']
['word', 'owrd', 'orwd', 'ordw', 'wrd', 'rowd', 'rodw', 'wrdo', 'rwdo', 'rdwo', 'rdw', 'wodr', 'owdr', 'odwr', 'odrw', 'wdr', 'dorw', 'dowr', 'dorw', 'wdro', 'dwro', 'drwo', 'drow']
```