Question 1. [4 marks]

Part (a) [1 mark] In the box below, write what this code prints:

```python
x = 5.5
y = x
x = x + 0.1
print(x, y)
```

Solution:
5.6 5.5

Part (b) [1 mark] In the box below, write what this code prints:

```python
t = 'nonunion'
prompt(len(t))
i = len(t) // 2
j = 0
while j <= i:
    print(t[j])
    j = j + 2
```

Solution:
8
n
n
n

Part (c) [2 marks] For each code fragment below, fill in the table with what it prints.

```python
s1 = 'swings'
s2 = 'slide'
```

<table>
<thead>
<tr>
<th>Code</th>
<th>What is printed</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(s1[2] + s2[3])</td>
<td>id</td>
</tr>
<tr>
<td>print(s1[1:2] + s2[2:])</td>
<td>wide</td>
</tr>
<tr>
<td>print(abs(len(s2) - len(s1)))</td>
<td>1</td>
</tr>
<tr>
<td>print(s2 in s1)</td>
<td>False</td>
</tr>
</tbody>
</table>
Question 2. [3 marks]

An employer gives a bonus to its employees based on how many items each employee sells. If the employee sells fewer than 10, they don’t get a bonus. If they sell 10 (inclusive) to 15 (exclusive) they get a $100 bonus, and if they sell at least 15, they get a $200 bonus.

Complete the following function according to the description above and its docstring below. Use the constants when appropriate.

```python
NOBONUS = 0
BONUS1 = 100
BONUS2 = 200

def bonus_amount(num_sold):
    """ (int) -> int

    Precondition: num_sold >= 0

    Return the employee bonus based on num_sold.

    >>> bonus_amount(7)
    0
    >>> bonus_amount(12)
    100
    >>> bonus_amount(16)
    200
    """

    if num_sold >= 15:
        return BONUS2
    elif num_sold >= 10:
        return BONUS1
    else:
        return NOBONUS
```

SOLUTION:
Question 3.  [11 marks]

Part (a)  [5 marks] Write the body of the following function according to its docstring description.

```python
def sort_letters(s1, s2):
    """ (str, str) -> tuple of (str, str)

    Precondition: s1 and s2 contain only lowercase letters,
    and len(s1) == len(s2)

    Return a tuple of two strings: the letters of s1 that are earlier in
    the alphabet than the letters at the corresponding position of s2, and
    the letters of s1 that are later in the alphabet than the letters at the
    corresponding position of s2. Omit corresponding letters that are the same.
    
    >>> sort_letters('ant', 'bib')
    ('a', 'nt')
    >>> sort_letters('knead', 'bread')
    ('n', 'k')
    """
    earlier = ''
    later = ''
    for i in range(len(s1)):
        if s1[i] < s2[i]:
            earlier = earlier + s1[i]
        elif s1[i] > s2[i]:
            later = later + s1[i]
    return (earlier, later)
```

SOLUTION:

```python
earlier = ''
later = ''

for i in range(len(s1)):
    if s1[i] < s2[i]:
        earlier = earlier + s1[i]
    elif s1[i] > s2[i]:
        later = later + s1[i]
return (earlier, later)
```
Part (b)  [3 marks] Write the body of the following function according to its docstring description. Do not use any str methods.

```python
def does_not_contain_char(s1, s2):
    """ (str, str) -> bool

    Return True iff s2 does not contain any characters from s1.
    ""

    >>> does_not_contain_char('hello', 'eat')
    False
    >>> does_not_contain_char('bye', 'snow')
    True

    SOLUTION:

    for ch in s1:
        if ch in s2:
            return False

    return True
```
Part (c)  [3 marks] Read the function header and body and then complete the docstring, including the type contract, description, and two examples that return different values.

```python
def mystery(s, ch):
    """ (str, str) -> bool

    Precondition: len(s) >= 1 and len(ch) == 1
    Return True iff the last character of s is ch, ignoring case.

    >>> mystery('hello', 'h')
    False
    >>> mystery('hello', 'o')
    True
    ""
```