**Question 1. [4 marks]**

Beside each code fragment in the table below, give the output. If the code would cause an error, write ERROR and give a brief explanation.

<table>
<thead>
<tr>
<th>Code</th>
<th>Output or Cause of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>winds = ['flute', 'oboe', 'bassoon']</code> <code>instruments = winds</code> <code>winds.append('clarinet')</code> <code>print(instruments[-1])</code></td>
<td>clarinet</td>
</tr>
<tr>
<td><code>brass = ['tuba', 'horn', 'bone']</code> <code>instruments = brass[0:2]</code> <code>instruments[0] = 'baritone'</code> <code>print(brass)</code></td>
<td>['tuba', 'horn', 'bone']</td>
</tr>
<tr>
<td><code>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</code> <code>print(times[1][1:])</code></td>
<td>[59, 'am']</td>
</tr>
<tr>
<td><code>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</code> <code>print(times[-2][0])</code></td>
<td>2</td>
</tr>
<tr>
<td><code>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</code> <code>print( times[0][0] &gt; 5 and times[1][1][1] &gt; 0)</code></td>
<td>False</td>
</tr>
<tr>
<td><code>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</code> <code>print(times[1][-1][0])</code></td>
<td>a</td>
</tr>
</tbody>
</table>
Question 2. [4 marks]

Read the function header and body and then complete the docstring. Give a meaningful function name, the type contract, the description, and two examples that return different values.

```python
def no_duplicate_letters(s):
    """ (str) -> bool
    Return True iff there are no duplicate letters in s.
    """
    seen = ''
    for ch in s:
        if ch in seen:
            return False
        else:
            seen = seen + ch
    return True
```

```python
>>> no_duplicate_letters('0123abcAa')
True
>>> no_duplicate_letters('AaA')
False
""
```

seen = ''
for ch in s:
    if ch in seen:
        return False
    else:
        seen = seen + ch

return True
Question 3.  [3 marks]

Complete the function below according to its docstring.

```python
def convert_to_integer(counts):
    """ (list of str) -> NoneType
    Replace each item in counts with its integer equivalent.
    Precondition: each item of counts is a string representation of a valid integer
    """
    for i in range(len(counts)):
        counts[i] = int(counts[i])
```

```python
>>> counts = ['1', '42', '0', '-4']
>>> convert_to_integer(counts)
>>> counts
[1, 42, 0, -4]
"""
```
Question 4.  [4 marks]

Complete the function below according to its docstring.

```python
def corrupted_text(corrupt, clean_up):
    """ (str, str) -> str

    Return a copy of corrupt with the following changes:
    All characters that are not letters, digits, or spaces
    are replaced by ''. 
    All characters in clean_up are replaced by '*'. 
    All other characters are left the same.
    
    Precondition: clean_up contains only alphanumeric characters
    
    >>> corrupted_text('Cor$\&r&^up&ted te**xt', 'jkqxyzJKQXYZ')
    'Cor r up ted te *t'
    >>> corrupted_text('aqn eKvil vxiruzrs dzid ttyhis!', 'jkqxyzJKQXYZ')
    'a*n e*vil v*iru*rs d*id tt*his '
    """
    result = ""
    for c in corrupt:
        if not c.isalnum() and not c == ' ':
            result += ' '
        elif c in clean_up:
            result += '*'
        else:
            result += c
    return result
```

```
**Question 5.** [5 marks]

Two students in a psychology class are playing a simple game. In each round of the game, they both place a secret bid on an item; the high bidder wins that item and must pay the average (the mean) of the two bids for the item. If there is a tie, neither player wins the item or pays any money.

Complete the following function according to the description above and the docstring below.

```python
def auction_average(player1_bids, player2_bids):
    ""
    (list of int) -> list of float
    ""
    Pre-condition: len(player1_bids) == len(player2_bids)
    
    Return a list of integers where the first element is the total amount
    of money to be paid by player 1 and the second is the amount to be paid
    by player 2. The bids made by each player are in player1_bids and player2_bids
    with one entry per item.
    
    >>> auction_average([6, 1, 7], [0, 0, 9])
    [3.5, 8.0]
    >>> auction_average([1, 10, 40, 100], [4, 10, 40, 0])
    [50.0, 2.5]
    ""

    cost1 = 0
    cost2 = 0
    
    for i in range(len(player1_bids)):
        if player1_bids[i] > player2_bids[i]:
            cost1 += (player1_bids[i] + player2_bids[i]) / 2
        elif player1_bids[i] < player2_bids[i]:
            cost2 += (player1_bids[i] + player2_bids[i]) / 2
    
    return [cost1, cost2]
```