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.

Code	Output or Cause of Error

`winds = ['flute', 'oboe', 'bassoon']`
`instruments = winds`
`winds.append('clarinet')`
`print(instruments[-1])`
	`clarinet`

`brass = ['tuba', 'horn', 'bone']`
`instruments = brass[0:2]`
`instruments[0] = 'baritone'`
`print(brass)`
	`['tuba', 'horn', 'bone']`

`times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]` `print(times[1][1:])`	`[59, 'am']`
`times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]` `print(times[-2][0])`	`2`
`times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]` `print( times[0][0] > 5 and times[1][1][1] > 0)`	`False`
`times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]` `print(times[1][-1][0])`	`a`

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.

```
def no_duplicate_letters(s):
    """ (str) -> bool
    Return True iff there are no duplicate letters in s.

    >>> no_duplicate_letters('01234abcA')
    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.

```
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

    >>> counts = ['1', '42', '0' , '-4']
    >>> convert_to_integer(counts)
    >>> counts
    [1, 42, 0, -4]
    """

    for i in range(len(counts)):
        counts[i] = int(counts[i])
```

Question 4. [4 marks]

Complete the function below according to its docstring.

```
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.

```
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]
```