Do not turn this page until you have received the signal to start.
(Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!

This midterm consists of 5 questions on 6 pages (including this one). When you receive the signal to start, please make sure that your copy is complete. Comments are not required except where indicated, although they may help us mark your answers. They may also get you part marks if you can’t figure out how to write the code. No error checking is required: assume all user input and all argument values are valid.

If you use any space for rough work, indicate clearly what you want marked.

# 1: _____/ 4  
# 2: _____/ 4  
# 3: _____/ 3  
# 4: _____/ 4  
# 5: _____/ 5  

TOTAL: _____/20
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>winds = ['flute', 'oboe', 'bassoon']</td>
<td></td>
</tr>
<tr>
<td>instruments = winds</td>
<td></td>
</tr>
<tr>
<td>winds.append('clarinet')</td>
<td></td>
</tr>
<tr>
<td>print(instruments[-1])</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>brass = ['tuba', 'horn', 'bone']</td>
<td></td>
</tr>
<tr>
<td>instruments = brass[0:2]</td>
<td></td>
</tr>
<tr>
<td>instruments[0] = 'baritone'</td>
<td></td>
</tr>
<tr>
<td>print(brass)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</td>
<td></td>
</tr>
<tr>
<td>print(times[1][1:])</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</td>
<td></td>
</tr>
<tr>
<td>print(times[-2][0])</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</td>
<td></td>
</tr>
<tr>
<td>print(times[0][0] &gt; 5 and times[1][1][1] &gt; 0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>times = [[4, 35, 'pm'], [2, 59, 'am'], [1, 10, 'am']]</td>
<td></td>
</tr>
<tr>
<td>print(times[1][-1][0])</td>
<td></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 (s):
    
    
    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

    >>> 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 '
    ""
```
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]
    """
```
.. builtins::
   :value: number
   :type: function

   abs(number) -> number
   Return the absolute value of the argument

   max(a, b, c, ...) -> value
   With two or more arguments, return the largest argument.

   min(a, b, c, ...) -> value
   With two or more arguments, return the smallest argument.

   print(value, ..., sep=' ', end='
') -> NoneType
   Prints the values. Optional keyword arguments:
   sep: string inserted between values, default a space.
   end: string appended after the last value, default a newline.

   int:
   int(x) -> int
   Convert a string or number to an integer, if possible. A floating point argument
   will be truncated towards zero.

   float:
   float(x) -> float
   Convert a string or number to a floating point number, if possible.

   str:
   str.count(sub[, start[, end]]) -> int
   Return the number of non-overlapping occurrences of substring sub in
   string S[start:end]. Optional arguments start and end are
   interpreted as in slice notation.

   str.find(sub[,i]) -> int
   Return the lowest index in S (starting at S[i], if i is given) where the
   string sub is found or -1 if sub does not occur in S.

   str.isalnum() -> bool
   Return True if all characters in S are alphanumeric
   and there is at least one character in S, False otherwise.

   str.isalpha() -> bool
   Return True if and only if all characters in S are alphabetic
   and there is at least one character in S.

   str.isdigit() -> bool
   Return True if and only if all characters in S are digits
   and there is at least one character in S.

   str.islower() -> bool
   Return True if and only if all cased characters in S are lowercase
   and there is at least one cased character in S.

   str.isupper() -> bool
   Return True if and only if all cased characters in S are uppercase
   and there is at least one cased character in S.

   str.lower() -> str
   Return a copy of S converted to lowercase.

   str.split([sep]) -> list of str
   Return a list of the words in S, using string sep as the separator and
   any whitespace string if sep is not specified.

   str.startswith(prefix) -> bool
   Return True if S starts with the specified prefix and False otherwise.

   str.strip() -> str
   Return a copy of S with leading and trailing whitespace removed.

   str.upper() -> str
   Return a copy of S converted to uppercase.