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 3 questions on 8 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.
- 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: _____/ 3
# 3: _____/11

TOTAL: _____/18
Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.
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)
```

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

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

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><code>print(s1[2] + s2[3])</code></td>
<td></td>
</tr>
<tr>
<td><code>print(s1[1:2] + s2[2:])</code></td>
<td></td>
</tr>
<tr>
<td><code>print(abs(len(s2) - len(s1)))</code></td>
<td></td>
</tr>
<tr>
<td><code>print(s2 in s1)</code></td>
<td></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
```

"""
Question 3. [11 marks]

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

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')
    """

>>> sort_letters('ant', 'bib')
('a', 'nt')
>>> sort_letters('knead', 'bread')
('n', 'k')
"""
Part (b)  [3 marks] Write the body of the following function according to its docstring description. Do not use any str methods.

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
"""
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.

def mystery(s, ch):
    
    """
    Precondition: len(s) >= 1 and len(ch) == 1
    """
    return s1[-1].upper() == ch or s1[-1].lower() == ch
Short Python function/method descriptions:

__builtins__:
  abs(x) -> number
  Return the absolute value of x.
  int(x) -> int
  Convert x to an integer, if possible. A floating point argument will be truncated
  towards zero.
  len(x) -> int
  Return the length of list, tuple, or string x.
  max(a, b, c, ...) -> object
  With two or more arguments, return the largest argument.
  min(a, b, c, ...) -> object
  With two or more arguments, return the smallest argument.
  print(value, ..., sep='', end='\n') --> NoneType
  Prints the values. Optional keyword arguments:
    sep: string inserted between values, default is a space.
    end: string appended after the last value, default is a newline.
  range([start], stop, [step]) -> list-like-object of int
  Return the integers starting with start and ending with stop - 1 with step
  specifying the amount to increment (or decrement). If start is not specified,
  the sequence starts at 0. If step is not specified, the values are incremented by 1.
  str(x) -> str
  Return an object converted to its string representation, if possible.

str:
  x in s --> bool
  Produce True if and only if x is in s.
  S.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.
  S.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.
  S.isalpha() -> bool
  Return True if and only if all characters in S are alphabetic
  and there is at least one character in S.
  S.isdigit() -> bool
  Return True if and only if all characters in S are digits
  and there is at least one character in S.
  S.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.
  S.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.
  S.lower() -> str
  Return a copy of S converted to lowercase.
  S.replace(old, new) -> str
  Return a copy of string S with all occurrences of the string old replaced
  with the string new.
  S.upper() -> str
  Return a copy of S converted to uppercase.