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:

```
k = 3
w = k
k = k + 1
print(k, w)
```

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

```
s = 'substitute'
print(len(s))
i = len(s) // 2
while i <= len(s):
    print(s[i - 1])
i = i + 2
```

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

```
s1 = 'elephant'
s2 = 'giraffe'
```

<table>
<thead>
<tr>
<th>Code</th>
<th>What is printed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print(s1[0] + s2[1])</code></td>
<td></td>
</tr>
<tr>
<td><code>print(s2[2:4] + s1[3:5])</code></td>
<td></td>
</tr>
<tr>
<td><code>print(abs(len(s2) - len(s1)))</code></td>
<td></td>
</tr>
<tr>
<td><code>print(not (s2 in s1))</code></td>
<td></td>
</tr>
</tbody>
</table>
Question 2. [3 marks]

A store is having a sale. A 10% discount is given for purchases from $50 (inclusive) to $100 (exclusive), and a 20% discount for purchases $100 and up.

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

```python
DISCOUNT1 = 0.10
DISCOUNT2 = 0.20

def apply_discount(dollar_amount):
    """ (float) -> float

    Precondition: dollar_amount >= 0

    Return dollar_amount with the appropriate discount applied.
    """

    >>> apply_discount(25.5)
    25.5
    >>> apply_discount(75.0)
    67.5
    >>> apply_discount(120.0)
    96.0
    """
```
Question 3.  [11 marks]

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

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

    Precondition: len(s1) == len(s2)

    Return a tuple of two strings: the characters of s1 that are the same as the characters
    at the corresponding position of s2, and the characters of s1 that are
    different from the characters at the corresponding position of s2.
    ""
    >>> sort_corresponding('hello!', 'heyya!')
    ('he!', 'llo')
    >>> sort_corresponding('bunk bed', 'buttered')
    ('bued', 'nk b')
    ""
```

Part (b) [3 marks] Write the body of the following function according to its docstring description. Do not use any str methods.

```python
def greater_than(s, d):
    ''' (str, int) -> bool

    Precondition: s contains only digits. d is a digit.

    Return True iff s contains a digit greater than d.

    >>> greater_than('1234', 3)
    True
    >>> greater_than('123', 5)
    False
    '''
```

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 s[0].upper() == ch or s[0].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='
') --> 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.