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.
- 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.
- You may use a pencil; however, work written in pencil will not be considered for remarking.

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

TOTAL: _____/18
Question 1. [3 marks]
Write the body of the following function according to its docstring description.

```python
def divisible_by_three(L):
    """ (list of int) -> bool
    Return True iff at least one item in L can be divided evenly by 3.
    Zero can be divided evenly by 3.
    >>> divisible_by_three([6])
    True
    >>> divisible_by_three([5, -7])
    False
    >>> divisible_by_three([1, 3, 5, -3, 9, 10])
    True
    """
```

Question 2. [2 marks]
Complete the example function calls by adding arguments that result in the return values shown. (For each example call, there are several correct answers, and providing any one of them will earn full marks.)

```python
def mystery(L):
    """ (list of list of int) -> list of int
    >>> mystery(
    [9, 9]
    >>> mystery(
    [2, 2, 5, 5, 8, 8]
    """
    result = []
    for item in L:
        result.append(item[0])
        result.append(item[0])
    return result
```
Question 3.  [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 (s1, s2):
    
    
    """
    Precondition: len(s2) == len(s1) and s1.isdigit() and s2.isdigit()
    """

    i = 0
    flag = True

    for ch in s1:
        ch2 = s2[i - 1]
        if ch != ch2:
            flag = False
            i = i + 1

    return flag
**Question 4.** [4 MARKS]

Write the body of the following function according to its docstring description.

```python
def ontario_area(area_codes):
    """ (list of int) -> NoneType
    Replace each number in area_codes with its corresponding area name:
    'Toronto', 'Ottawa', or 'Other'. Area codes 416 and 647 belong
    to 'Toronto', and area codes 343 and 613 belong to 'Ottawa'. All other area
    codes belong to area 'Other'.
    """

>>> area_codes = [500, 416, 647, 613]
>>> ontario_area(area_codes)
>>> area_codes
['Other', 'Toronto', 'Toronto', 'Ottawa']
"""
Question 5.  [5 marks]

Write the body of the following function according to its docstring description.

def encode(message, encodings):
    """ (str, list of str) -> int

    Preconditions:
    - len(message) >= 1 and len(encodings) >= 1
    - each string in encodings has length 2
    - every character in message appears as the first character in exactly
      one of the encoding strings.
    - the second character in each encoding string is a character in the
      range '1' to '9'.

    Based on the information in encodings, convert each character in message
    to a digit and return those digits all together as a single integer.

    >>> encode('g', ['a1', 'h2', 'g3', 'j4', 'y5', 'n6'])
    3
    >>> encode('code', ['e1', 'c2', 'p3', 'd4', 'o5', 'n6'])
    2541
    """

Short Python function/method descriptions:

__builtins__:

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.replace(old, new) -> str
        Return a copy of string S with all occurrences of the string old replaced with the string new.

list:
    x in L --> bool
        Produce True if and only if x is in list L
    L.append(object) -> NoneType
        Append object to end of list L.
    L.extend(iterable) -> NoneType
        Extend list L by appending elements from the iterable. Strings and lists are
        iterables whose elements are characters and list items respectively.