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>
</table>
| toys = ['ball', 'car', 'doll']
stuff = toys[0:2]
stuff[0] = 'phone'
print(toys)                                                        |                          |
| toys = ['ball', 'car', 'doll']
stuff = toys
stuff.pop()
print(toys)                                                        |                          |
| days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7], ['Fri', 'Dec', 6]]
print(days[1][1][1:2])                                              |                          |
| days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7], ['Fri', 'Dec', 6]]
print(days[-1][-2])                                                 |                          |
| days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7], ['Fri', 'Dec', 6]]
print(days[0][0] == 'Thurs' and days[4] == [])                       |                          |
| days = [['Mon', 'Oct', 21], ['Wed', 'Nov', 7], ['Fri', 'Dec', 6]]
print(days[-1][1:])                                                  |                          |
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):
    ""
    if len(s) == 0:
        return True
    first_lower = s[0].lower()
    for ch in s:
        if ch.lower() != first_lower:
            return False
    return True
    ""
```

Question 3.  [3 MARKS]

Complete the function below according to its docstring.

```python
def make_absolute(nums):
    """ (list of number) -> Nonetype
    Replace each item in nums with its absolute value.
    >>> nums = [1, -4.5, 0.2, -6]
    >>> make_absolute(nums)
    >>> nums
    [1, 4.5, 0.2, 6]
    """
```
Question 4. [4 marks]

Complete the function below according to its docstring.

```python
def new_math_test(old_test, operators):
    """ (str, str) -> str

    Return a copy of old_test with the following changes:
    All digits are replaced by ' '.
    All symbols (not letters, digits, or spaces) in operators
    are replaced by '_'
    All other characters are left the same.
    """

>>> new_math_test('x^3 + 3x^2 - 17', '+-')
'x^ _ x^ _ '

>>> new_math_test('Find the integral of 4x*e^2x + 19', '')
'Find the integral of x*e^ x + '

"""
```
**Question 5.**  [5 marks]

Two parties are running in many elections in regions throughout a country. In each region, whichever party receives the most votes wins the region, and whichever party wins the most regions will get to form the government.

Complete the following function according to the description above and the docstring below.

```python
def election_results(party1_votes, party2_votes):
    """ (list of int, list of int) -> list of int

    Pre-condition: len(party1_votes) == len(party2_votes)

    Return a list of integers representing the number of regions won by each party. The first element is the number of regions won by party 1, the second is the number won by party 2 and the third is the number of ties. The number of votes for each region are in the lists party1_votes and party2_votes, with one entry per region.

    >>> election_results([5, 2, 8], [0, 0, 9])
    [2, 1, 0]
    >>> election_results([17, 13, 40, 100], [18, 10, 40, 0])
    [2, 1, 1]
    """
```
__builtins__:
    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:
    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.isalnum() -> bool
    Return True if all characters in S are alphanumeric
    and there is at least one character in S, False otherwise.
    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.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.
    S.startswith(prefix) -> bool
    Return True if S starts with the specified prefix and False otherwise.
    S.strip() -> str
    Return a copy of S with leading and trailing whitespace removed.
    S.upper() -> str
    Return a copy of S converted to uppercase.