CSC 108H5 F 2013 Midterm Test
Duration — 50 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____ **First Name:** _____

Lecture Section: L0101/L0102 Instructor: Dan

---

*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 page). *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: _____/10

TOTAL: _____/18

Question 1. [4 MARKS]

Part (a) [2 MARKS] In the table below are 4 calls to the function `returns_what`. Beside each call, write the returned value and its type.

```
def returns_what(x):
    ''' (int) -> object
    '''

    if 100 >= x and x > 3:
        if x % 2 == 0:
            return x / 2
        else:
            return 19 - x
    elif x < 3 or x > 100:
        if abs(x) <= 5:
            return x ** 3
        else:
            return x * 2
```

Call	Return Value	Return Type
`returns_what(24)`		
`returns_what(17)`		
`returns_what(-2)`		
`returns_what(3)`		

Part (b) [1 MARK] Write the output of the code below in the box. If the code causes an error, please write **error** instead of giving output.

```
x = 3
x = x + 1 + x
print(x)
```

Part (c) [1 MARK] Fill in the box with Python code that will make the program behaviour match the comments. You may **not** make any other changes to the code or add code outside the box.

```
def is_fun(chore):
    ''' (str) -> bool
    'laundry' is not fun. 'taxes' is not fun. Every other chore is fun.
    Return True if and only if chore is fun according to these rules.
    chore is guaranteed to be lowercase.'''

    return
```

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.

```
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. [10 marks]

Part (a) [5 marks] Complete the function according to its docstring.

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

    Return a version of old_test with the following changes:
      Each digit is replaced by ' '.
      Each character in operators is replaced by '_'.
      All other characters are left the same.
      operators is guaranteed not to contain any spaces or digits.

    >>> 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 +   '
    '''
```

Part (b) [5 marks] Complete the function according to its docstring.

```
def change_elements(lst, change):
    '''(list of int, list of bool) -> NoneType
    lst and change are lists of the same length.
    For exactly those elements in change that are True,
    change the corresponding element of lst to 99. Make no other changes to lst.
    For example, if change were [True, False, True], then
    lst[0] and lst[2] would be set to 99.
    '''
```

[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.]

[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.]

Last Name: _____ **First Name:** _____

Short Python function/method descriptions:

```
__builtins__:
  input([prompt]) -> str
    Read a string from standard input; return that string with no newline. The prompt string,
    if given, is printed without a trailing newline before reading.
  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='\n') -> 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.
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.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.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.
```