CSC 148H5 F 2016 Test 1
Duration — 50 minutes
Aids allowed: none

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

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

☐ Lecture Section: L0101 Instructor: Dan Zingaro

---

*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 test consists of 4 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.
If you use any space for rough work, indicate clearly what you want marked.

# 1:	_____/ 6
# 2:	_____/ 5
# 3:	_____/ 4
# 4:	_____/ 5
TOTAL:	_____/20

Question 1. [6 MARKS]

Write the following function that removes all consecutive duplicates from a stack.

Consider stack `1 1 1 2 1 3 3`. When called on this stack, the function would change the stack to `1 2 1 3`. Another example is in the below docstring.

Assume that the `push`, `pop`, and `is_empty` stack methods are available. Do not call any method besides these three.

```
def remove_dups(s: Stack) -> None:
  '''Modify s so that consecutive duplicates are removed.

  >>> s = Stack()
  >>> s.push(1)
  >>> s.push(2)
  >>> s.push(2)
  >>> remove_dups(s)
  >>> s.pop()
  2
  >>> s.pop()
  1
  '''
```

Question 2. [5 MARKS]

Define the **depth** of an element in a list as the number of list indexes required to access the element. For example, in the list `[10, [20, [[30]]], 40]`, the depth of 10 is 1, depth of 20 is 2, depth of 30 is 4, and depth of 40 is 1.

Write the following function according to its docstring.

```
def max_depth(lst):
  '''(list possibly with nesting) -> int

  Return the maximum depth of any element in lst.
  If lst has no elements, return 0.
  lst may be nested arbitrarily.

  >>> max_depth([10, [20, [[30]]], 40])
  4
  >>> max_depth([])
  0
  >>> max_depth([[], 10])
  1
  '''
```

Question 3. [4 MARKS]

Explain in plain English the purpose of the following mystery function. (Remember: this means that we want the overall purpose of the code, **not** a line-by-line description of what the code does.)

```
def mystery(lst):
  q = Queue()
  ret = []
  for item in lst:
    q.enqueue(item)
  while not q.is_empty():
    element = q.dequeue()
    if isinstance(element, list):
      for element2 in element:
        q.enqueue(element2)
    else:
      ret.append(element)
  return ret
```

Question 4. [5 MARKS]

Write the following function so that it satisfies its docstring. Your code *must be recursive.* (A **precondition** is something that you can assume to be true when your function is called.)

```
def divide(s: str, chr: str) -> str:
  '''
  Precondition: chr is a character that appears in string s.
  Precondition: Every character in s is unique and lowercase.

  Return a new string where all characters that are alphabetically less
  than chr come first, then chr, then all characters that are
  alphabetically greater than chr.

  >>> divide('jkldcf', 'f')
  'dcflkj'
  >>> divide('xyzdabc', 'd')
  'abcdzyx'
  '''
```

[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.
```