CSC 148H5 F 2016 Test 2
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 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.
If you use any space for rough work, indicate clearly what you want marked.

# 1:	______/ 4
# 2:	______/ 6
# 3:	______/ 6
TOTAL:	______/16

Question 1. [4 marks]

The **postorder** traversal for a **binary search tree** t is as follows:
3, 2, 1, 4, 5, 7, 6

Draw t.

Question 2. [6 marks]

Here is a `BTNode` class:

```
class BTNode:
  """A node in a binary tree."""

  def __init__(self: 'BTNode', item: object,
               left: 'BTNode' =None, right: 'BTNode' =None) -> None:
    """Initialize this node.
    """
    self.item, self.left, self.right = item, left, right
```

Write the following function.

```
def longest_path(t: BTNode) -> list:
  """Return a Python list containing the values in a longest path of t.
  If there are multiple longest paths, return a list of one of them.

  >>> b1 = BTNode(7)
  >>> b2 = BTNode(3, BTNode(2), None)
  >>> b3 = BTNode(5, b2, b1)
  >>> longest_path(b3)
  [5, 3, 2]
  """
```

Question 3. [6 marks]

Here is a Node class:

```
class Node:

  """Node in a linked list"""

  def __init__(self: 'Node', value: object, next: 'Node'=None) -> None:
    """Create Node self with data value and successor next."""
    self.value, self.next = value, next
```

And here is a LinkedList class:

```
class LinkedList:

  """Collection of Nodes to form a linked list"""

  def __init__(self: 'LinkedList') -> None:
    """Create empty LinkedList"""
    self.front, self.back, self.size = None, None, 0
```

Write the following **method** for the LinkedList class. The method **modifies the linked list** so that each node value appears twice in a row. For example, if your linked list were
`1->0->9->9`
then your method would change the linked list to
`1->1->0->0->9->9->9->9`

```
  def repeat_items(self: 'LinkedList') -> None:
    """Repeat each item in LinkedList self."""
```

[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.]

[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.
```