Duration: **180 minutes**
Aids Allowed: ***None***

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

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This exam consists of 9 questions on 20 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete.*

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

MARKING GUIDE

# 1:	_____/ 15
# 2:	_____/ 10
# 3:	_____/ 25
# 4:	_____/ 10
# 5:	_____/ 10
# 6:	_____/ 10
# 7:	_____/ 5
# 8:	_____/ 5
# 9:	_____/ 10
TOTAL:	_____/100

Good Luck!

For the duration of this exam, you can assume that you have access to the following Python classes:

```
class LLNode(object):
    '''A Node in a singly-linked list'''

    def __init__(self, data):
        '''(LLNode, object) -> NoneType
        Create a new node to hold data
        '''
        self.data = data
        self.link = None


class DLLNode(object):
    '''A Node in a doubly-linked list'''

    def __init__(self, data):
        '''(DLLNode, object) -> NoneType
        Create a new node to hold data
        '''
        self.data = data
        self.next = None
        self.prev = None


class BTNode(object):
    '''A Node in a binary tree''

    def __init__(self, data):
        '''(BTNode, object) -> NoneType
        Create a new node to hold data
        '''
        self.data = data
        self.left = None
        self.right = None
```

Question 1. [15 marks]

Implement the following ADT:

`LetterQueue:` A container for holding alphabetical letters, with the following public methods:

- `__init__()`: initialize a new empty queue
- `enqueue(L)`: insert the letter L onto the end of the queue
- `dequeue_vowel()`: remove and return the next vowel in the queue
- `dequeue_consonant()`: remove and return the next consonant in the queue
- `size()`: returns the number of elements in the queue
- `num_caps()`: returns the number of capital letters in the queue
- `num_lower()`: returns the number of lower-case letters in the queue
- `is_empty()`: returns `True` iff the queue is empty

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 2. [10 marks]

For the `LetterQueue` ADT defined in the previous question, design a series of test cases to optimally cover all testing of the ADT.Your code does not have to be in standard `UnitTest` style, but your assertions should use the standard assertion calls, such as: `AssertEqual, AssertTrue, AssertRaises, AssertIsInstance, etc`. You do not have to complete an exhaustive set of tests, but you should choose a small number of representative tests for optimal coverage.

Purpose	Assertion	Test Type
Test that a newly created LetterQueue is empty	`my_lq = LetterQueue()` `assertTrue(my_lq.isEmpty)`	Black Box

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 3. [25 marks]

You have been hired to write a program to manage the data of a small community college. The dean has sent you the following list of requirements:

- We have a number of courses, with course codes like "ABC123", each course has exactly one instructor, and may have zero or more students
- Lecture courses meet on a particular day of the week in room (all rooms have numbers 0-500)
- Reading courses don't have a specific time/place to meet, but they still have students and a teacher
- We have three types of people in the school, students (we need to know their date of birth, student number and name), faculty (we need to know their name, date of birth, office number and a salary), and staff (we need to know their name, date of birth and salary).
- I should be able to create/modify courses/people and any of their fields. If someone puts in bad info, it should give me an error explaining what I did wrong.
- When I print a course, I'd like to see it's info, plus all students taking it, and when I print a student, I'd like to see their info and a list of all the courses they're taking, when I print a faculty or staff member, I just want to see their info.
- We've hired a programmer from our own course to do the actual implementation of the code, they are familiar with Python, but have no idea how to structure a program. So you must provide them with the class/inheritance structure, and method details, including DocStrings, as well as a few internal comments per method to explain to an experienced programmer how to complete the methods, but you do not need to provide the actual method bodies (hope you read all the way through before you started coding).

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 4. [10 marks]

Complete the following function according to its DocString. For full marks, your code *must* be recursive:

```
def add_nested(L):
    """(nested list of numbers) -> number
    Return the sum of all the numbers in the nested list L.
    REQ: L and all of the components of L are either numbers, or nested lists of numbers
    >>>add_nested([[],[[],[]]])
    0
    >>>add_nested([[], 1, [1, 1]])
    3
    >>>add_nested([2, 3, [4, [[[5]], 6]], 7])
    27
    """
```

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 5. [10 marks]

Complete the following function according to its DocString.

```
def reverse_list(L):
    """(LLNode) -> LLNode
    Given the head of a linked list L, return the head of a linked list with the same
    elements as L, but in reverse order
    """
```

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 6. [10 marks]

Complete the following function according to its DocString.

```
def second_largest(root):
    """(BTNode) -> object
    Return the vale of the second largest node in the binary search tree rooted at root
    REQ: root is the root of a properly formed binary search tree, with >= 2 nodes
    """
```

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 7. [5 marks]

Draw the Heap (the abstract node based representation, not the list implementation) resulting from the following series of operations (assume you are using a max-heap (largest value at the root), and starting with an empty Heap):

- Insert 10
- Insert 20
- Insert 30
- Insert 15
- Insert 5
- Insert 50
- Remove-Head

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 8. [5 marks]

Consider running the `MergeSort` algorithm on the following list:

`[9,5,7,3,2,1,6,8]`

What sub-lists will `merge` be called on over the course of the sort?

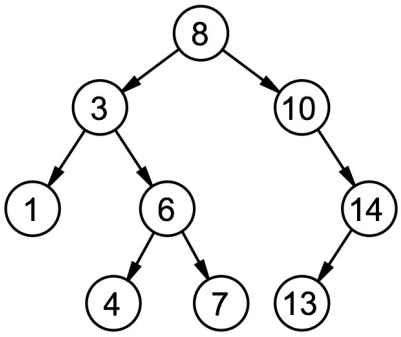
For example, if `merge` will be called twice, once to merge the lists `[a,b,c]` and `[d,e,f]`, and once to merge the lists `[x,y,z]` and `[w]`, your answer would be:

```
merge([a,b,c],[d,e,f])
merge([x,y,z],[w])
```

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 9. [10 marks]

Consider the following Splay Tree:



a) [5 marks] Draw the tree resulting from the operation `insert(15)`

b) [5 marks] Draw the tree resulting from the operation `delete(3)` **performed on the original tree, not the modified tree from part a)**

End of Exam: Total Marks = 100