Name: _____

This exam is a worth a maximum of 150 points and you have 75 minutes to complete it.

1. **Abstract Data Types**
   (a) (5 points) Give at least two examples of why using **Abstract Data Types** can be useful when programming. Use appropriate terminology when possible.
   (b) (5 points) "Getters and setters" are frequently used in object-oriented languages. Are they necessary to read and write values for an object's local variables when using Python? Why are they used?
   (c) (5 points) What method must be be overloaded so that a user of your abstract data type can use iteration such as `for x in a`? What data type must it return?

2. **Generators**
   Create the following generators with functions using `yield`:
   (a) (5 points) A generator that starts at 5 and will always return a multiple of 5.
   (b) (5 points) A generator that will return even numbers in order from 0 to 20 and then stop.

3. **Functional programming**
   (a) (5 points) Why are functions such as `map`, `reduce`, and `filter` useful tools when programming for clustered computers with "Big Data"?
   (b) (5 points) Use a `filter` to take a list of numbers and return only the even numbers.
   (c) (5 points) Use a `map` that takes a list of words and returns a list of word-lengths.

4. **Counting**
   (a) (5 points) What is the largest positive integer value you can represent with 4 bits?
   (b) (5 points) How many possible strings can be created that are exactly 6 letters long? Assume there are 26 possible letters, 'a' through 'z', all lowercase. You can express this quantity concisely using exponential notation.
   (c) (5 points) Assume 20 different cities. If you calculate the distance from each city to every other city, how many total distances do you have to compute?

5. **Computer Memory**
   (a) (5 points) What is it called when two different symbols (variable names) point to the same memory location?
   (b) (5 points) In most languages arrays are typically homomorphic, but Python lists are considered polymorphic. What does this mean, and how is this implemented internally by the language?
   (c) (5 points) What is the value of `b.price` after the following code has run?

```
class Product:
    def __init__(self, price)
        self.price = price

a = Product(3.00)
b = Product(5.00)
c = b
b.price = 4.00
print c.price
```

(d) (5 points) What is the value of c after the following code has run?

```
a = "fire"
b = "water"
c = a
a = "wind"
print c
```

6. **Big O notation**

If an algorithm at the worst case requires the following number of iterations, express the complexity of that algorithm in Big O notation. Assume n is the size of the input data.

(a) (5 points) $log_2(n) + 300 * n * log_2(n)$

(b) (5 points) $x^n + 300x^3$

7. (20 points) **Searching**

Write an function `find_element` that will find an element in a list in log_n time. Return `None` if the element does not appear in the list, otherwise return the index position of the element. Use the following function definition, but replace `pass` with your implementation.

```
def find_element(theList, theElement):
   pass
```

8. **Hashing**

What are the following time complexities of using a hash?

(a) (5 points) What is important to consider when creating a hash function?

(b) (5 points) What is the best case time complexity?

(c) (5 points) What is the worst case time complexity?

9. **Arrays vs. Linked Lists**

Create a table like below on your answers page. What is the time complexity for doing the following operations on an array and linked list? Replace the ? with your appropriate answer. Assume the linked list has a tail node.

	Dynamic Array	Linked List
Searching	?	?
Indexing	?	?
Insert to the beginning	?	?
Append to end	?	?
Remove from middle	?	?

10. (30 points) **Linked Structures**

You are to create a *Queue* data structure in which a user can insert a value on one end of a list (with a `push` operation) and then have a value returned on the other end (with a `pop` operation). Assume that the size of your queue is unbounded in the number of elements inserted. Replace the sections that say `pass` with your code. You may use any local variables you find convenient such as a head or tail reference in your solution. Do not use Python lists but instead only use only `Node` objects. Return `None` if pop is used on an empty Queue.

```
class Node (object):

    def __init__(self, value):
        self.value = value
        self.next = None

    def getValue(self):
        return self.value

    def setValue(self, value):
        self.value = value

    def getNext(self):
        return self.next

    def setNext(self, nextNode):
        self.next = nextNode


class Queue (object):

  def __init__(self):
        pass

    def push(self, value):
        pass

    def pop(self, value):
        pass

if __name__ == "__main__":
    q = Queue()
    q.push(5)
    q.push(7)
    q.push(9)

    print q.pop()
    print q.pop()
```

The result of running the following code in the main function results in printing 5 and 7. There will be a 9 left in the list.