ECE 241
Practice Exam for Midterm 1

Prof. M. Zink

Name: _____

ID Number: _____

	Maximum	Achieved
Question 1	16	
Question 2	18	
Question 3	24	
Question 4	18	
Question 5	24	
Total	100	

This exam is closed book, closed notes. No electronic devices (including calculators) are allowed. Be concise, but show your work. Write legibly. When writing code, please indent appropriately and give your variables meaningful names.

Time: 120 minutes.

Question 1 (16 points):
Answer the following questions regarding a queue class with the following structure:

```
class Queue:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def enqueue(self, item):
        …

    def dequeue(self):
        …

    def size(self):
        return len(self.items)
```

a) Assume the enqueue() and dequeue() methods are implemented correctly. What is the output of the following program? (4 points)

```
q=Queue()
q.enqueue(4)
q.enqueue('dog')
q.enqueue(True)
q.enqueue(2.8)
q.dequeue()
q.dequeue()
```

None
[4]
[4, "dog"]
[4, "dog", True]
[4, "dog", True, 2.8]
["dog", True, 2.8]
[True, 2.8]

b) Provide an implementation of the `enqueue()` method. (6 points)

```
self.append(item)
return self
```

c) Provide an implementation of the `dequeue()` method. (6 points)

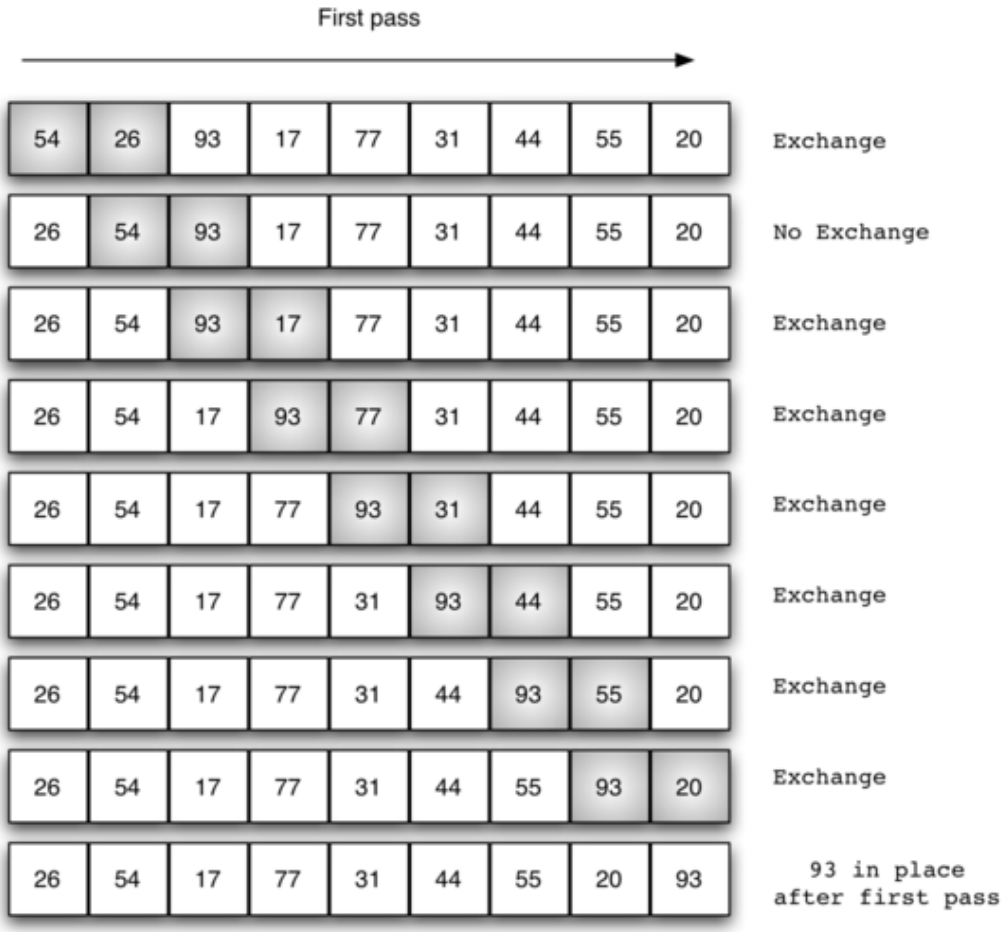
```
self.pop(0)
return self
```

Question 2 (18 points):

a) The listing below shows the partial implementation of the insertion sort algorithm. Complete the missing code segments in lines 4, 5, 7, 8, 9 indicated by … (8points)

```
1: def insertionSort(alist):
2:    for index in range(1,len(alist)):
3:
4:      currentvalue = alist[…]
5:      …
6:
7:      while position>0 and …:
8:          …
9:          …
10:
11:     alist[position]=…
12:
13: alist = [54,26,93,17,77,31,44,55,100]
14: insertionSort(alist)
15: print(alist)
```

b) Which sorting algorithm is shown in the image below? (4 points)



Bubble sort

c) Determine the complexity (O()) of the following code examples (6 points)

```
def tripple_loop(x, y, z):
     for i in range(x):
          for j in range(y):
               for k in range(z):
                    print(i, j, k)
```

O(xyz)

```
def mult_increment(n):
     for i in range(0, n, i=i * 2):
          print('Start', i)
          for j in range(i):
               print(i, j)
     print('----------------------------')
```

Question 3 (24 points):
The following sequence is recursively defined as

$$F_n = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ F_{n-1} + F_{n-2} & if\ n > 1 \end{cases}$$

and implemented in Python as:

```
def F(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return F(n-1)+F(n-2)
```

a) F(3) = F(2) + F(1) = F(1) + F(0) + F(1) = 1 + 0 + 1 = 2
F(5) = F(4) + F(3)
= F(3) + F(2) + F(2) + F(1)
= F(2) + F(1) + F(1) + F(0) + F(1) + F(0) + F(1)
= F(1) + F(0) + F(1) + F(1) + F(0) + F(1) + F(0) + F(1)
= 1 + 0 + 1 + 1 + 0 + 1 + 0 + 1 = 5

a) For the computation of F(3) and F(5), list a step by step function calls of F() to get the final answer. (8 points)

b) How many times is F() called for the computation of F(3)? (2 point)
5

c) How many times is F() called for the computation of F(4)? (2 points)
9

d) Write a method that recursively calculates the factorial of n, which is defined as n! = n * (n-1)! (8 points)

```
def Factorial (n):
if n == 0:
return 1

else:
return Factorial(n - 1) * n
```

e) Given the following code that implements the binary search algorithm fill out the table for the list `testlist = [18,17,13,12,11,8,5,3]` (5 Points)

testlist.sort()

```
def binarySearch(alist, item):
    first = 0
    last = len(alist)-1
    found = False

    if first<=last:
        midpoint = (first + last)//2
```

```
        if alist[midpoint] == item:
            found = True
        else:
            if item < alist[midpoint]:
                alist = alist[0:midpoint]
                return binarySearch(alist,item)
            else:
                alist = alist[midpoint+1:]
                return binarySearch(alist,item)
```

item = 13

```
    return found
```

iteration	first	last	midpoint
1	0	7	3
2	0	3	1
3	0		
4	0		

found

Question 4 (18 points):

Answer the following questions regarding links and objects. Consider the following code:

```
class SomeObject:
    def __init__(self, value):
    self.x    = value
    self.link = None

a = SomeObject(3)
b = SomeObject(4)
c = SomeObject(5)

# Problem a)
a.link = b
b.link = a
c.link = c
print(a.link.x, b.link.x, c.link.x)

# Problem b)
a.link = b.link
b.link = c.link
c.link = a.link
print(a.link.x, b.link.x, c.link.x)

# Problem c)
a.link.x = 6
b.link.link.x = 7
c.link.link.link.x = 8
print(a.link.x, b.link.x, c.link.x)
```

b, a, c

a -> b ->c

a, c, a

a -> a
b -> c
c -> a

a-> 6
b -> c -> 7
c -> a -> 6 -> 8

a) What is the output of the first print instruction? Provide an illustration of the objects, their values, and their link relationships. (Only showing the output will not give you full points.) (6 points)

b) What is the output of the second print instruction? Provide an illustration of the objects, their values, and their link relationships. (Only showing the output will not give you full points.) (6 points)

c) What is the output of the third print instruction? Provide an illustration of the objects, their values, and their link relationships. (Only showing the output will not give you full points.) (6 points)

Question 5 (24 points):

Answer the following questions regarding linked lists. For this question, you can assume that the linked list is singly linked and only has one pointer to the beginning of the list. Each list element should store an integer that is provided by the user. Below the class Node and the initial part of the class Ordered list are given.

```
class Node:
    def __init__(self,initdata):
        self.data = initdata
        self.next = None

    def getData(self):
        return self.data

    def getNext(self):
        return self.next

    def setData(self,newdata):
        self.data = newdata

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


class OrderedList:
    def __init__(self):
        self.head = None
```

index based 0

```
def search (self, item):
    curr = self.head
    cnt = 0
    while curr:
        if curr.getData() == item:
            return cnt
        curr = curr.getNext()
        cnt += 1
    return -1
```

```
def remove (self, item):
    item_idx = self.search(item)
    if item_idx > 0:
        curr = self.head
        for _ in range (item_idx):
            curr = curr.getNext()
        nxt = curr.getNext()
        nxt = nxt.getNext()
        curr.setNext(nxt)
```

a) Based on the class Node and the initial part of class Ordered list, implement a method `search(self, item)` that searches for an integer value in the list and returns the position of the value if found or -1 if not found (4 points)

b) Based on the class Node and the initial part of class Ordered list, implement the method `remove(self,item)` that removes an item (in the form of an integer value) if it exists in this sorted list (10 points)

c) Write a method `reverseOrder()` as part of the Ordered list class, that reverses the ordwer of the ordered list. E.g, if the current list is 1,2,3,4,5, a new list 5,4,3,2,1 should be created. (10 points)

```
def reverseOrder(self):
    curr = self.head
    prev = None
    while curr:
        nxt = curr.getNext()
        curr.setNext(prev)
        prev = curr
        curr = nxt

    return prev
```