Data Structures - Test 1

Question 1. (10 points) Determine the theta notation $\Theta(\cdot)$ for the following Python code.

```python
for i in xrange(n):
    for j in xrange(i):
        sum = i + j
# end for j
# end for i
```

Question 2. (10 points) Suppose a $\Theta(n^2)$ algorithm takes 10 seconds when $n = 1,000$. How long would you expect the algorithm to run when $n = 10,000$?

Question 3. (15 points) For the two implementations of fibonacci given below, explain why fibA is so much slower than fibB.

```python
def fibA(n):
    if n <= 1:
        return n
    else:
        return fibA(n-1) + fibA(n-2)

def fibB(n):
    fibs = [0, 1]
    for i in range(2, n+1):
        fibs.append(fibs[i-1] + fibs[i-2])
    return fibs[n]
```
Question 4. (5 points) What is the difference between unit testing and integration testing?

Question 5. (15 points)
a) In the following recursive binary search code, what would be a precondition on the binarySearch function?

```python
def binarySearch(myList, target):
    """Returns the position of the target in myList or -1 if not found""
    def binarySearchHelper(myList, target, first, last):
        print "first is", first, "last is", last
        if first > last:
            return -1           # -1 indicates target not found in myList
        else:
            midpoint = (last+first)/2
            if myList[midpoint] == target:
                return midpoint
            elif target < myList[midpoint]:
                return binarySearchHelper(myList, target, first, midpoint-1)
            else:
                return binarySearchHelper(myList, target, midpoint+1, last)
    return binarySearchHelper(myList, target, 0, len(myList)-1)
```

b) Show the output of the following program which calls binarySearch. (INCLUDE the output of the debugging print statement in the binarySearchHelper function)

```python
aList = [10, 20, 30, 40, 50, 60, 70, 80]
print "The list is: ", aList
target = 50
location = binarySearch(aList, target)
if location == -1:
    print target, "NOT found"
else:
    print target, "FOUND at index", location
```

Output of the above program which calls binarySearch:
Question 6. (25 points) Consider the following `AltStack` class that uses an Array to store the items in the stack. The “top” item on the stack is always stored at index 0. (NOTE: this is different from the `ArrayStack` class of section 14.4)

a) Complete the theta notation $\Theta(\cdot)$ for each stack methods of the above `AltStack` implementation: (Let us define "n" as the # items in the stack)

<table>
<thead>
<tr>
<th>Method</th>
<th><em>init</em> (constructor)</th>
<th>push(item)</th>
<th>pop()</th>
<th>peek()</th>
<th>len()</th>
<th>isEmpty()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theta notation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Assume that the array size DOES NOT grow during the `push` method, but has a fixed physical capacity from the `_init_` constructor. What would be the precondition on the `push` method.

c) Write the code for the `push` method of the `AltStack` class.
```python
def push(self, newItem):
    """Inserts newItem at the top of stack."""
```
Question 7. (20 points) Consider the following AltLinkedStack class which uses the Node class (from the text and listed above) to dynamically create storage for a new item added to the stack. Conceptually, an AltLinkedStack object would look like the below picture. (NOTE: this is different from the LinkedStack class in section 14.4)

(a) Complete the theta notation \( \theta() \) for each stack methods of the above AltLinkedStack implementation: (Let us define "n" as the # items in the stack)

<table>
<thead>
<tr>
<th>Method</th>
<th>_init_ (constructor)</th>
<th>push(item)</th>
<th>pop()</th>
<th>peek()</th>
<th>len()</th>
<th>isEmpty()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theta notation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Write the code for the push method of the AltLinkedStack class.

```python
def push(self, newItem):
    """Inserts newItem at the top of stack."""
```