4/9/08 **Lab 11 Binary Heap** Name:_____

Objectives: To understand how a binary heap works, and how to use a heap to sort a list.

Activity 1: The text's version of a binary heap is available by copying the folder P:\810-063-CSIII\LABS\LAB_11. Open and run the binaryHeap.py file in IDLE which contains the text's class BinaryHeap that implementing a "min heap". Now, test the BinaryHeap class with the following:

```
h = BinaryHeap()
h.insert(5)
h.delMin()
h.insert(5)
h.insert(10)
h.insert(3)
h.insert(1)
h.delMin()
```

a) Did you get what you expected for the results of **both** the delMin() method calls?

b) Use `h.heapList` to look at the current list of the `BinaryHeap`. What is odd about the list?

c) After you play with and study the BinaryHeap class, what do you think is causing the problem?

After you have answered the above questions, raise your hand and explain your answers.

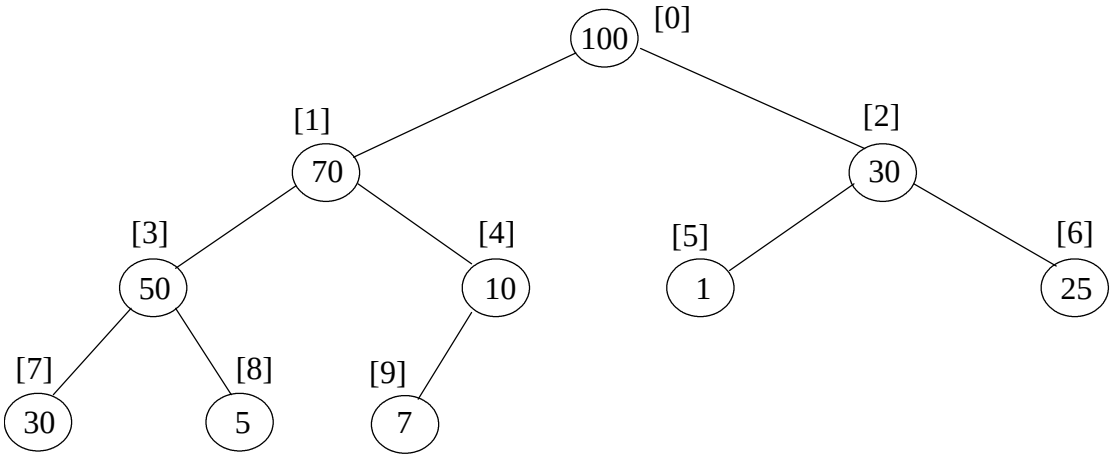
Activity 2: Fix the binary heap code from the text.

After you have tested your code, raise your hand and demonstrate that it works correctly.

Lab 11 Binary Heap

Name:_____

Activity 3: The text's code implements a "min heap" by using a list to store the items in the heap. It also wastes the zeroth index in the list. I want you implement a "max heap" and NOT waste the zeroth index in the list, i.e.,



For the above "max" heap, the list indexes are indicated in [ ]'s. For a node at index *i*, what is the index of:
a) its left child if it exists:

b) its right child if it exists:

c) its parent if it exists:

You MaxHeap class should have methods: insert(key), findMax(), delMax(), isEmpty(), size(), buildHeap(list), and increaseKey(key) similar to those found on page 224 of the text for a min heap.

After you have implemented and tested your "max" heap, raise your hand and demonstrate that it works correctly.

Activity 4: Use your "max" heap implementation in Activity 1 to implement a sorting function that sorts in ascending order (i.e., from smallest to largest).

After you have implemented your "heap-based" sort, raise your hand and demonstrate that it works correctly.

NOTE: If you complete all of the activities within the lab period, you do not need to hand anything it. However, if you need to finish activities outside of the lab period, then hand in printouts of code and output for activities not completed during the lab period.

Lab 11 Binary Heap

Name:_____

```
class BinaryHeap:
    def __init__(self):
        self.heapList = [0]
        self.currentSize = 0

    def percUp(self,i):
        while i > 0:
            if self.heapList[i] < self.heapList[i/2]:
               tmp = self.heapList[i/2]
               self.heapList[i/2] = self.heapList[i]
               self.heapList[i] = tmp
            i = i/2

    def insert(self,k):
        self.heapList.append(k)
        self.currentSize = self.currentSize + 1
        self.percUp(self.currentSize)

    def percDown(self,i):
        while (i * 2) <= self.currentSize:
            child = i * 2
            mc = self.minChild(i)
            if self.heapList[i] > self.heapList[mc]:
                tmp = self.heapList[i]
                self.heapList[i] = self.heapList[mc]
                self.heapList[mc] = tmp
            i = mc

    def minChild(self,i):
        if i*2 > self.currentSize:
            return -1
        else:
            if i*2 + 1 > self.currentSize:
                return i*2
            else:
                if self.heapList[i*2] < self.heapList[i*2+1]:
                    return i*2
                else:
                    return i*2+1

    def delMin(self):
        retval = self.heapList[1]
        self.heapList[1] = self.heapList[self.currentSize]
        self.currentSize = self.currentSize - 1
        self.percDown(1)
        return retval

    def buildHeap(self,alist):
        i = len(alist) / 2
        self.currentSize = len(alist)
        self.heapList = [0] + alist[:]
        while (i > 0):
            self.percDown(i)
            i = i - 1
```