CIS 291 – Data Structures in C++ - Spring 2005
Week 12 Lab Exercise
Week 12 Lab Exercise due: by END of LAB on Tuesday, April 12th

Purpose:
To become more familiar with one implementation of a complete tree and of a heap by reading their code in order to answer questions about them; to practice with the algorithms for insertion into and removal from a heap.

Week 12 Lab Exercise
In CIS 291 HW #10, you are using a provided **heap** template class to implement **heapsort**. This **heap** class happens to be implemented with the help of a template class **complete_tree**, which is an array-based implementation of a complete binary tree.

Answer the following questions on paper; after coming up with your initial answers, you may discuss them with another student before getting your answers checked, if you wish. When you are ready, put your name on the "Next:" list on the board so that your answers can be checked.

1. Consider the implementation of the **complete_tree** template class that you are using in HW #10.

(a) List the names of the **accessor/observer** methods provided by **complete_tree**.

____

____

____

(b) List the names of the **modifier** methods provided by **complete_tree**.

____

____

____

(c) List the **types AND names** of the private **data fields** used within **complete_tree**.

____

____

(d) What private methods are declared within **complete_tree**? List their names.

____

____

(e) Give the formula that this **complete_tree** implementation uses for determining the index of the **parent** of the current node (assuming the index of the current node is stored within **current_index**).

____

(f) Give the formula that this **complete_tree** implementation uses for determining the index of the **left child** of the current node (assuming the index of the current node is stored within **current_index**).

____

(g) Give the formula that this **complete_tree** implementation uses for determining the index of the **right child** of the current node (assuming the index of the current node is stored within **current_index**).

____

2. Now consider the implementation of the **heap** template class that you are using in HW #10.

(a) List the names of the **accessor/observer** methods provided by **heap**.

____

(b) List the names of the **modifier** methods provided by **heap**.

____

(c) List the **types AND names** of the private **data fields** used within **heap**.

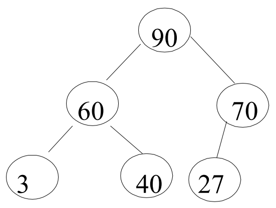
____

(d) What private methods are declared within **heap**? List their names.

____

3. In lecture, we discussed the basic algorithms for inserting into and removing from a heap. For this problem, you are going to show that you understand these basic algorithms.

Consider the following heap:



(a) Consider what should happen in order for **100** to be added to this heap. You should have in your lecture notes the basic steps for adding to a heap --- you can also refresh your memory by looking at the appropriate place within the **heap** template class implementation.

On the back of one of this lab assignment's pages or on a separate sheet of paper, write **3(a)** and then **draw** the stages that this heap goes through during the process of adding **100** to this heap. (You must give a snapshot of each change to this heap that occurs in the process of adding 100 to it. The final picture will how the heap "ends up".)

(b) ASSUME that **(a)** has been done. Now we want to add **75** to the heap.

On the back of one of this lab assignment's pages or on a separate sheet of paper, write **3(b)** and then **draw** the stages that this heap goes through during the process of adding **75** to this heap. Again, you must give a "snapshot" of each change to the heap that occurs.

(c) ASSUME that **(a)** and **(b)** have been done. Now add **14** to the heap, writing **3(c)** and then **drawing** the stages that the heap goes through during this process

(d) ASSUME that **(a) - (c)** have been done. Now **REMOVE** the the maximum value in this heap, writing **3(d)** and then **drawing** the stages that the heap goes through during this process.

What value is returned by this **remove_max** operation? ________________________

(e) ASSUME that **(a) - (d)** have been done. Now, remove the maximum value in this heap again, writing **3(e)** and then **drawing** the stages that the heap goes through during this process.

What value is returned by this **remove_max** operation? ________________________

(f) ASSUME that **(a) - (e)** have been done. Now, remove the maximum value in this heap one more time, writing **3(f)** and then **drawing** the stages that the heap goes through during this process.

What value is returned by this **remove_max** operation? ________________________