ICS 202 HOMEWORK#3 (Term 071)

Note: The report for this homework must be typed; however you may **neatly** draw the required diagrams by hand.

Question 1 [12 points: 2 + 10] Complexity analysis of recursive algorithms

(a) Write the recurrence relation for the running time **T(n)** of the function **f(n)** given below.

```
public static int f(int n){
  if(n == 1)
     return 2;
  else
     return f(n/2) + f(n/2) + g(n);
}

public static int g(int n){
  int sum = 0;
  for(int k = 1; k <= n*n; k++)
     sum += k;

  return sum;
}
```

DO NOT SOLVE THE RECURRENCE RELATION

$$T(n) = \begin{cases} a & \text{if} \quad n = 1 \\ 2T(n/2) + n^2 + c & \text{if} \quad n > 1 \end{cases}$$

(b) The recurrence relation for the running time **T(n)** of the function **f(n)** given is:

$$T(n) = \begin{cases} a & \text{if} \quad n = 1 \\ 4T(n/2) + n^2 + c & \text{if} \quad n > 1 \end{cases}$$

Solve the recurrence relation by the method of ***unrolling and summing*** to find the big-O complexity of **f(n)**.

Use appropriate summation formulae from ICS202 lecture notes.

Question 2 [20 points: 13 + 7]

A hospital uses a **MinHeap** to maintain a priority queue of patients who want to see a doctor. Each patient is assigned an integer priority. Each object in the priority queue is an **Association** object with the priority, an **Integer** object, as the **key** and the **value** being a patient object (The **Patient** class is provided).

(a) **[13 points]** Write a method: **public void modifyPriority(int newpriority, Patient obj)** of the **BinaryHeap** class that will modify the priority of patient **obj** to **newpriority** and then rearrange the priority queue such that the corresponding **Association** object is in its proper location in the priority queue. Your method must throw an appropriate exception if there is no Association object in the priority queue with patient **obj** as its value.

(b) **[7 points]** Write a menu driven program with the following menu:

1. Initialize the Priority Queue with (priority, patient) Association objects. (Use the given **patients.txt**)
2. Insert a (priority, patient) Association object in the priority queue.
3. Delete the (priority, patient) object with the highest priority (i.e., lowest priority value) from the Priority Queue.
4. Modify the priority of a patient with a given ID number to a given priority.
5. Display the Priority Queue.
6. Exit

Note: Each line of **patients.txt** contains: Priority, ID number, name, and gender of a patient. The ID numbers are unique; but the priorities are not.

Question 3 [20 points: 16 + 4]

(a) **[16 points]** Write an instance method:
`public MySearchableContainer duplicatedKeys()` of the **BinaryTree** class which calls a **recursive** private helper instance method of the **BinaryTree** class. The helper method returns a **MySearchableContainer** that contains unique keys that are duplicated (i.e., that appear more than one time) in the invoking **BinaryTree** object.

Example: if the Integer keys of the BinaryTree are 50, 10, 50, 30, 20, 40, 30, 60, 30, 18, 2, 7, 18 then the returned container will contain the keys: 50, 30, and 18.

Note:

(i) You methods must be general i.e.; they must work for any given **BinaryTree** instance.
(ii) Your methods must not use any looping statements, iterator, static or instance variables.

(b) **[4 points]** Write program to test the **duplicatedKeys( )** method.

Note:

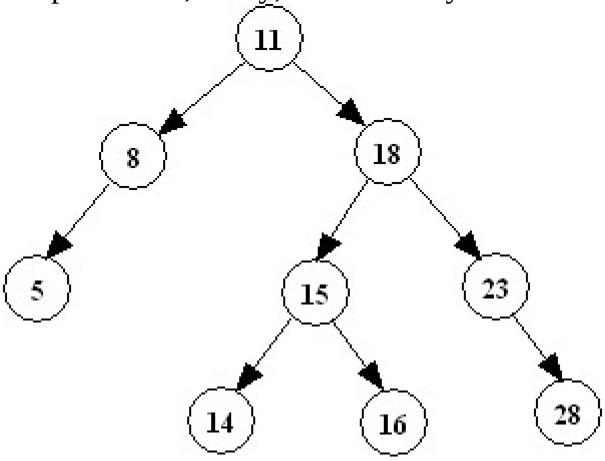
(i) To test the **duplicatedKeys()** method, your program can use the given **BinaryTreeBuilder** class or it can create directly a **BinaryTree** instance that contains some duplicated keys.
(ii) To use **Binary TreeBuilder** class, you are required to read the document **Building_a_BinaryTree.doc**

Question 4 [15 points: 5 + 10]

(a) **[5 points]** Show the result (final tree only) of converting the array given below into a min-heap by the bottom-up approach:

15	10	13	9	6	7	8	17	8	5

(b) **[10 points]** Draw the intermediate trees, the final tree, and mention the rotations performed, if any, when the key **17** is inserted in the AVL tree given below:

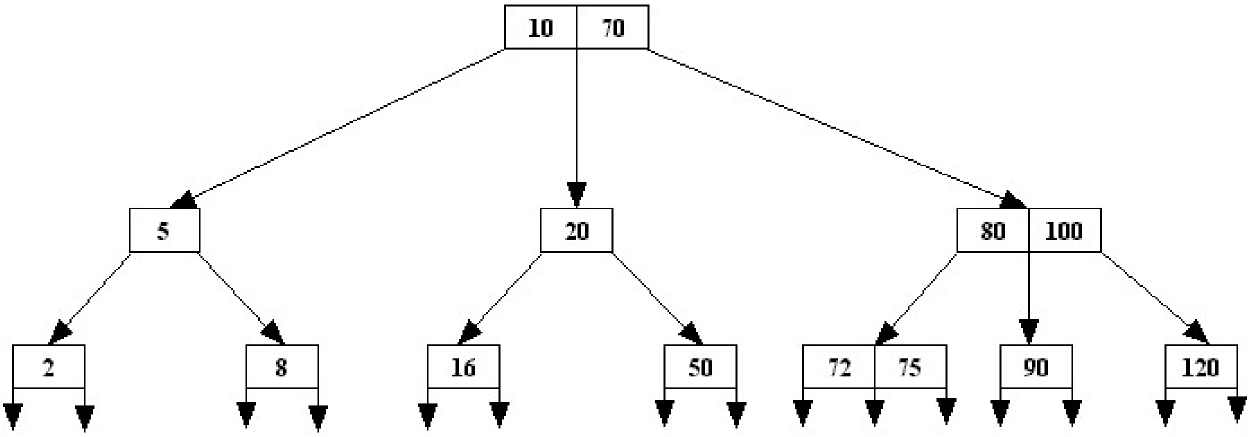


Question 5 [18 points: 8 + 10]

(a) Draw the final B-Tree and the intermediate trees that results in inserting the following keys in an initially empty B-tree of order 3:

7, 14, 4, 20, 8, 22, 32, 6, 5, 10, 12

(b) **[10 points]** Draw the B-tree that results in deleting the key **50** from the following B-tree of **order 3**:



Question 6 [15 points: 6 + 3 + 1 + 2 + 3] Huffman Coding

A file contains the characters: **BTAMMAMMSAEBMMBBETAAMTTAMB**

(a) Draw a Huffman code tree for encoding the textfile. **Show all intermediate and the final tree and the way these trees are arranged in a MinHeap priority queue.**

Note: Create a unique Huffman code tree by using the following guidelines:

1. If two or more initial one-node trees have the same frequency; arrange them in the MinHeap priority queue in alphabetically increasing order.
2. Whenever two trees are dequeued from the priority queue; the first dequeued tree becomes the left subtree of the merged tree, and the second dequeued tree becomes the right subtree.
3. A merged tree **t** must be inserted **after** all trees in the MinHeap priority queue with the same frequency as that of **t**.
4. The left edge of each node in the final tree is assigned **0** and the right edge of each node is assigned **1**

Note: NO GRADE WILL BE GIVEN FOR THIS QUESTION IF THESE 4 GUIDELINES ARE NOT FOLLOWED

(b) Write the Huffman codeword of each character in a table similar to the following: **(0.5 for each codeword)**

character	Huffman codeword
A	
B	
E	
M	
T	
S	

(c) **[1 point]** Encode the message: **BASEET**

(d) **[2 points]** Decode the message **001001100010111011000** if possible

(e) **[3 points]** Calculate the number of bits to transmitt the encoded file **x**, show the details of your calculations

character	**A**	**B**	**E**	**M**	**T**	**S**	**TOTAL**
codeword							
Codeword bits							
Character frequency							
Character frequency * Codeword bits							