Design and Analysis of Algorithms: Homework 3 (50 pts)

1. (a) (5 points) Illustrate the execution of the heap-sort algorithm on the following sequence: (2, 5, 16, 4, 10, 23, 39, 18, 26, 15). Show the contents of the heap and the sequence at each step of the algorithm. Indicate upheap or downheap bubbling where appropriate.

   (b) (5 points) Illustrate the execution of the bottom-up construction of a heap (like in Figure 2.49) on the following sequence: (2, 5, 16, 4, 10, 23, 39, 18, 26, 15, 7, 9, 30, 31, 40).

2. (10 points) Let T be a heap storing n keys. Give the **pseudocode** for an efficient algorithm for printing all the keys in T that are smaller than or equal to a given query key x (which is not necessarily in T). You can assume the existence of a $O(1)$-time *print(key)* function. For example, given the heap of Figure 2.41 and query key $x = 7$, the algorithm should report 4,5,6,7. Note that the keys do not need to be reported in sorted order. Your algorithm should run in $O(k)$ time, where k is the number of keys reported.

3. Use the table below to convert a character key to an integer for the following questions.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
Key	0	1	2	3	4	5	6	7	8	9	10	11	12
Letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Key	13	14	15	16	17	18	19	20	21	22	23	24	25

   (a) (5 points) Give the contents of the hash table that results when the following keys are inserted in that order into an initially empty 13-item hash table: $(E_1, A, S_1, Y, Q, U, E_2, S_2, T, I, O, N)$. Use $h(k) = k \mod 13$ for the hash function for the k-th letter of the alphabet (see above table for converting letter keys to integer values). Use linear probing.

   (b) (5 points) Give the contents of the hash table that results when the same keys are inserted in that order into an initially empty 13-item hash table. Use $h(k) = k \mod 13$ for the hash function for the k-th letter of the alphabet (see above table for converting letter keys to integer values). Use double hashing and let $h'(k) = 1 + (k \mod 11)$ be the secondary hash function.

4. (a) (4 points) Insert into an initially empty binary search tree items with the following keys (in this order): 30, 40, 23, 58, 48, 26, 11, 13. Draw the tree after all insertions. Include a few intermediate stages.

   (b) (4 points) Remove from the binary search tree in Figure 3.7(a) the following keys (in this order): 32, 65, 76, 88, 97. Draw the tree after **each** removal.

   (c) (2 points) A different binary search tree results when we try to insert the same sequence into an empty BST in a different order. Give an example of this with at least 5 elements and show the two different binary search trees that result.

5. (10 points) Let T be a binary search tree, and let x be a key. Give an efficient algorithm for finding the smallest key y in T such that $y > x$. Note that x may or may not be in T. Explain why your algorithm has the running time it does.