ECS 60 Practice Midterm #1 Key

1 (30 points) We expect quite short answers to these questions, i.e., two or three sentences each.

a) (15 points) In programming it is often best to copy an existing function that is similar to your new needs, and then alter the copy to suit the new requirements. If I wanted a function to print out a range of numbers in reverse order, which of the four traversals functions would you copy as a basis for the new function?
InorderTraversal().

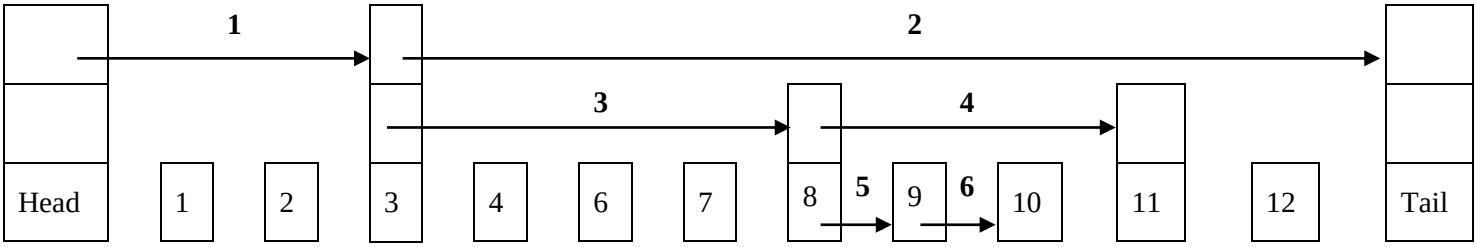
b) (15 points) This question applies to the timetest program. If File5.dat contained 5,000,000 insertions, how would you change your program to be able to accommodate File5.dat? Would any ADT take more than ten times as long on File5.dat as File1.dat (which contained 500,000 inserts)?
Both the QueueAr and StackAr constructors would have to be initialized to some value larger than 5,000,000. The skip list would take slightly more than 10 times as long.

2 (46 points) The registrar keeps track of all the students (simply using their SIDs) on the wait lists for each course (specified by their CRNs). As you know, the wait lists must keep track of the order of arrival of each request. There are two reasons to be removed from a wait list: 1) The first person on the list is assigned to the class; or 2) A student withdraws his/her request for the class associated with wait list. Insertions and deletions are frequent. Once a quarter instructors have the wait list printed for their courses, but the registrar never tries to print out all of the wait list students at the university. Assume there are M = 1000 courses, and an average of N = 10 students on each wait list. Describe the data structure(s) you would use to satisfy these specifications most time efficiently. Justify you choices, using big-O notation where appropriate. You may not use hash tables.

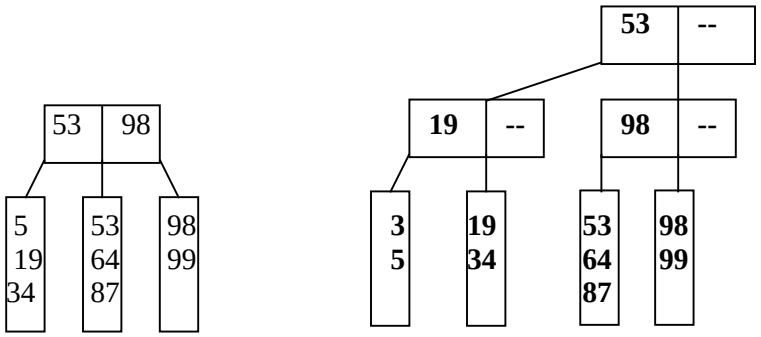
First, note that there are only M*N = 10000 entries with 9 chars per entry. This is small enough to fit in RAM, so there is no need for a B-Tree. Second, note that the number of students waiting for each course are so small that the difference between log n and n is negligible. Finally, because of the need to delete from positions other than the first position, queues are eliminated.

Based on these facts I would use a splay tree of lists to store the information. The splay tree keys would be the CRNs, and each node would contain a list of waiting maintained in FIFO order. Insertions would be placed at the end of the list and take amortized O(1 + log M) time, with a tail pointer if a linked list implementation is used. Deletions for enrolled students would take amortized O(1 +log M) for linked list implementations and O(N + log M) for array implementations. Deletions for students that just give-up would take amortized O(N + log M) time for both implementations. To print a wait list would take amortized O(N + logM) time for both implementations.

3 (10 points) In the following Skip List, draw and number sequentially the path that is checked in order to find(10).



4 (15 points) Given the following 3-ary BTree with L = 3, draw the BTree that would exist after an insert(3).

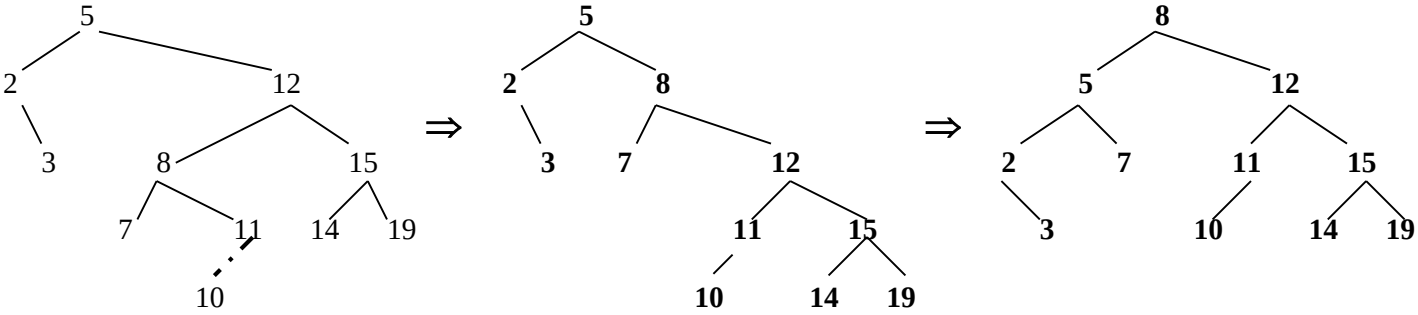


5. (14 points) Assuming double hashing, a hash function h_1 of key % TableSize, a hash function of h_2 of key % 5 + 1, and an initial TableSize of 7, show the hash table after each of the operations. Rehash when the load factor would become greater than 0.5. Remember that the table size must always be a prime. Also fill-in the TableSize column.

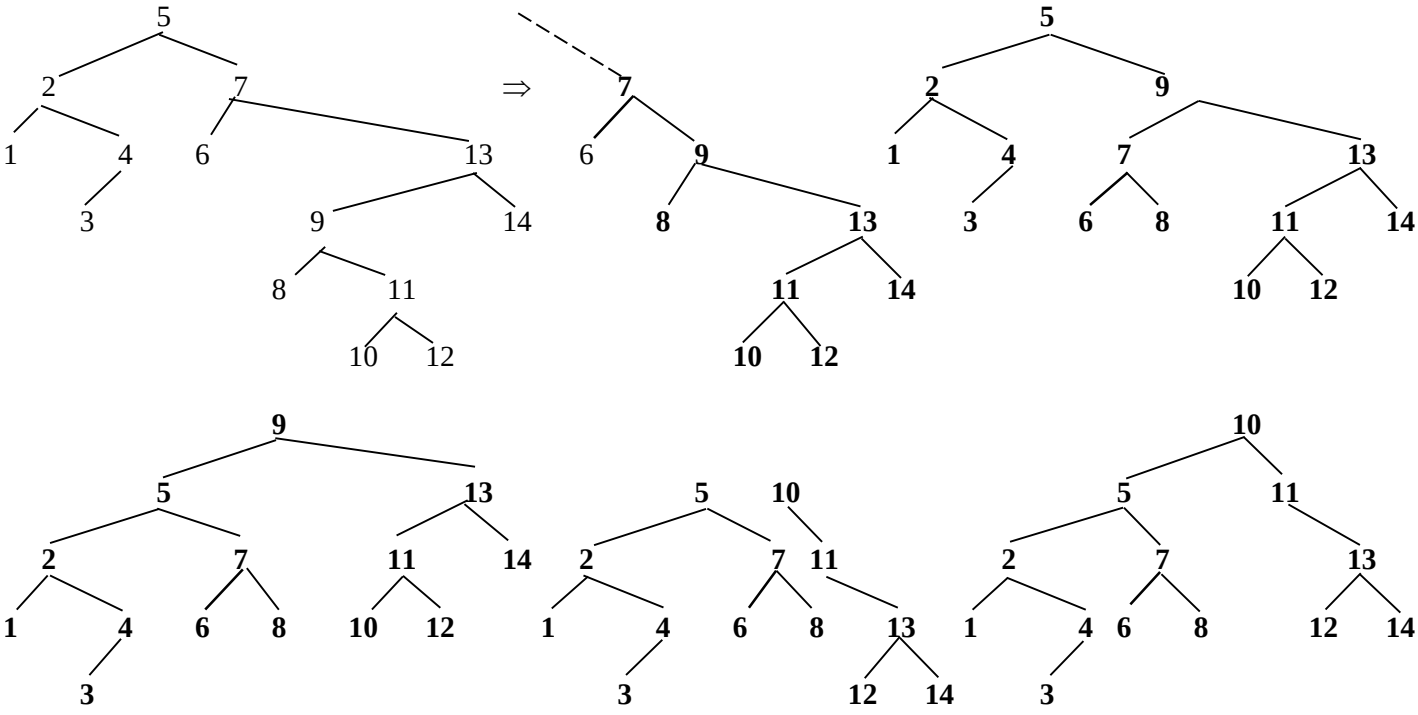
Points assigned for the changed item(s) only.

Operation	TableSize	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Pts
insert 38	7				**38**														1
insert 21	**7**	**21**			**38**														**1**
insert 3	**7**	**21**			**38**	**3**													**3**
insert 20	**17**				**3**	**21**	**20**			**38**									**5**
delete 21	**17**				**3**		**20**			**38**									**1**
insert 2	**17**			**2**	**3**		**20**			**38**									**1**
insert 36	**17**			**2**	**3**	**36**	**20**			**38**									**2**

6. (15 points) Given the following AVL tree, draw a representation of the tree after an insert(10) operation has been executed. (You may wish to write intermediate tree(s) to ensure partial credit.)



7. (20 points) Given the following splay tree, what would the tree look like after the deletion of 9 using a normal splay. (You may wish to write intermediate tree(s) to ensure partial credit. Rather than copying parts of the tree that remain the same from the previous intermediate tree, you may simply use dashed lines from the appropriate node to indicate that either the ancestors or descendants remain the same from the previous intermediate tree. However, your final tree must be complete, and not use any dashed lines.)



4 **points for each intermediate tree.**