ECS 60 Practice Midterm #1 (150 points)

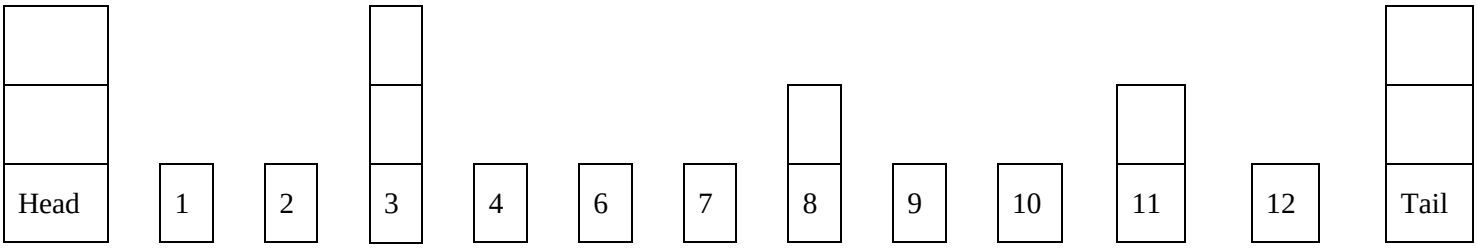
1 (30 points) We expect quite short answers to these questions, i.e., two or three sentences each.

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?

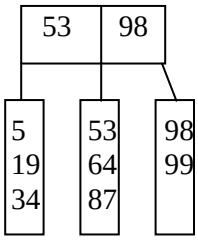
b) (15 points) This question applies to the timetest program. If File5.dat contained 2,500,000 insertions, how would you change your program to be able to accommodate File5.dat? Would any ADT take more than ten times as long on File5.dat as File1.dat (which contained 250,000 inserts)?

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. You may not use hash tables. Justify you choices, using big-O notation where appropriate. You may not use a hash table.

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



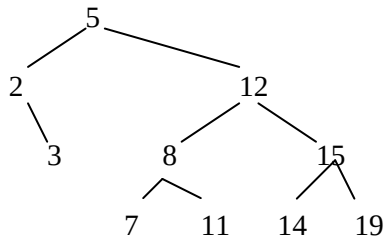
4 (15 points) Given the following 3-ary BTree with L = 3, draw the BTree that would exist after an insert(3).



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. Quadratic probing, linear probing, and double hashing are all possible on the test.

Operation	Table Size	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
insert 38	7																			
insert 21																				
insert 3																				
insert 20																				
delete 21																				
insert 2																				
insert 36																				

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 11 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.)