Nicholas Harris

40111093

Assignment 2 – COEN 352

1. Insertion sort will run faster than selection sort if all keys identical. This is because selection sort will traverse the whole array for each value, for n items in an array this will take O(n2) time. While insertion sort will simply traverse the array once as an array of identical keys is already sorted, leading to O(n) time.
2. You would first shift through the array, comparing the first and second cards. If they are out of order, swap and move the top card to the end of the deck. If they are in order, move the top card to the bottom. Repeat until you can shift through every element without swapping the first two.
3. The best case of shell sort is 0(n), when the data is completely sorted, and the gap is a constant value.
4. printKthElement(Node head, int k){

static int count = 0;

if((list.getSize() – k - count) == 0){

return head.data;

}

if(count++ == k);

printKthElement(head.next, k);

}

1. a. Merge sort is suited for very large inputs as you do the sorting on smaller and smaller sub arrays, which are stored in memory and are then copied back into the original. It has a consistent speed over different sizes of inputs while quick sort is only good for small inputs.

b. Merge sort can’t be done in place as it requires additional memory because of the sub-arrays used to sort the data.

1. An unordered array will be most effective, if there only a small amount of remove max operations, ordering an array isn’t necessary and maintaining a heaps condition by moving through the min/max values isn’t necessary either. The insert operation is in constant time for unordered and the search for max is O(n).
2. We use quick sort on sequences A and B so it is easier to compare each of the elements to see if they contain the same keys between set A and set B. If we were to do this on the unsorted sequences, we would need to compare a single element of A to at most each element of B. Sorting the sequences and skipping duplicates allows you to move through the sequences linearly. The time complexity of the sorting is O(nlogn) and the complexity of determining if they contain the same sets is O(n). O(nlon) + O(n) = O(nlogn).