1) Describe the workings of your code and challenges that you encountered during your program development. [ 3 points ]

2)  Program listing with line numbers. [ 10 points ]

3)  Create test cases and report their output. [ 7 points]

Problem 1:

1. Workings of the code and challenges.

Problem 2:

1. Workings of the code and challenges.

Problem 3 (Sorting):

1. Quicksort is a divide and conquer style algorithm designed to break a larger sorting problem into a subset of smaller problems. I employed two different recursive solutions to Quicksort in my solution. My partition-based Quicksort uses a “partition” function to divide the entire array into the smaller sub arrays. High and low values swap as the index positions move inwards towards the center of the array. After swapping values on the first sub list the partition returns the position of the index. In doing so the problem becomes increasingly smaller as more of the sub lists become sorted. The other method of Quicksort that I am using does not use a partition. Instead the pivot position is selected from the middle of the array. On subsequent recursive calls to quicksort the algorithm shifts the index position which emulates the behavior of the partition. My biggest issue with this algorithm was managing my base case. Since there are several calls to quicksort it is possible to lose track of the case which stops all recursion. This was challenging to work through since infinite recursion prevents the system from providing much information as to where things went wrong. I found that using partition simplified this issue greatly, as the this allowed me to separate the code into very smaller testable chunks of code.

Problem 4:

1. Workings of the code and challenges.

Problem 5 (Dictionary):

1. I chose to create a dictionary using a linked list of linked lists. My dictionary enforces unique keys and values. It is capable of loading key-value pairs from a file and storing them in a container. The container can be queried providing a “key” word to the “get” method. The get method will return a KeyNode which has all the associated meanings (values) of for the keyword or a null value if the key is not present in the dictionary. The major issues I dealt with on this assignment had mainly to do with testing for the presence of a key/value inside one of the dictionary nodes. This was due to how java tests for equality between objects. The solution ended being to use the built-in string equals method to ensure that the string values of the nodes were equal or not. I spent the most time testing the corner cases for my linked list. Since java doesn’t use pointers in the traditional sense it was interesting learning how use the pseudo reference of storing the next value.