Assignment 2- Ciaran Gray

Q1.

i)

A well defined recursive function is a function that calls upon itself within the function and uses its own previous term to repeatedly calculate subsequent terms. It has two properties, it has at least one value where the function returns the best case scenario where the function can return a result without having to make any other recursive calls, like a "stopping point", or more formally known as the base case. For the find1() function, the base case is where the array arrA[] only has one element within it. The other property a recursive function has is when it has a recursive case where the function reduces the input values while calling upon itself within the function to try and get closer to the base case with each recursion, this is known as the "reduce" property. In the find1() function, this is seen when the current element is smaller then the size-2 element of the array, in which case the function calls upon itself and reduces the size and uses the corresponding size-1 element as the current element.

ii)

Line	Cost	NumTimes	NumTimes*Cost	Total
2				
3				
4	1	N		
5	1	1		
6				
7	1	N		
8				
9				
10				
11	1	N		
12				
13				

2.i)

The merge function essentially takes an array of random order, splits the elements of the array up into smaller arrays until an element is left on it's own, and then reverses the elements back into the bigger arrays, each time re-arranging the array in order, until every element is back in the original array in the correct order.

For example, lets say that the array arrA[] = {6, 3, 8, 4, 1, 9, 5, 2} and an array size of 8. The lb and up integers represent the lower and upper bound of the array. When we look at the mergeSort function, in the if statement on line 5, if the upperbound of the array is bigger then the lowerbound of the array, which it will be for the first call as lb is set to 0 and ub is size-1, so in this case 7, the mid integer is set to 3.5, which rounds to 4., the middle of the array. Then the function is called again, with the ub being replaced with mid on line 7, called again on line 8 with the lb being set to mid+1 and the ub being set back to 7. This is to split the array into 2. This splits the array continuously until there are many sublists created of 1

element each. Then the merge function is called. It accepts three integers. On line 17 a new array is created to store the newly organised array of numbers. The for loop on line 20 sets int i to the lowebound, in this case 0, and int j to mid plus 1, in this case 1 It runs for when 0 is less or equal to 2 and when 2 is less or equal to the ub, which is 7. This loop will merge the elements back together in order with the smallest one first in the arrC array, or if the if statement on line 21 is false, the order will stay the same. Then, on line 29 and 32, two while loops will copy any remaining elements back into the arrC array, if there are any. Finally, on line 35, the temporary holding array arrC ‘s elements are copied back into the original arrA array, in order and the arrC array ir ready to take in the next subset of elements. So the order at the end should look like arrA = {1, 2, 3, 4, 5, 6, 8, 9}.

3.

To display the number of recursion calls, comparisons and data swaps I created 3 integers and increased their value every time a recursion, comparison or data swap happened.