2) a) Ternary(dataset, Lowest point, highest point, key):

if ( highest point – lowest point > 0)

check dataset[lowest value] with key :

return lowest key

check dataset[highest value] with key:

return if highest key

first midpoint = lowest point + (highest point – lowest point)/3

second midpoint = first midpoint + ( highest midpoint - first midpoint) /3

check dataset[first midpoint] with key :

return first midpoint

check dataset[second midpoint] with key:

return if second midpoint

check if (key < dataset[mid first])

return Ternary(dataset, lowest point, first midpoint,key)

check if (key > dataset[second midpoint])

return Ternary(dataset, second midpoint, highest point,key)

return ternary(dataset, first midpoint, second midpoint,key)

b) ternary tree reoccurrence equation:

My pseudo program does 12 operations (6 comparisons, 2 addition, 2 subtraction, 2 divide) and recursively calls Ternary () with 1/3 of the dataset that is passed in. therefore:

T(n) = T(n/3) + 12 is my equation –; T(n/3) being the recusive call and 12 representing the number of operations to complete this step.

c)

else

return error NOT FOUND

3)

a) min\_max(list)

if size(list) = 1

return (list[0],list[0])

else

firsthalfLowHigh = min\_max(firsthalfList)

secondhalfLowhigh = min\_max(secondHalfList)

if(firsthalfLowHigh[0] < secondhalfLowHigh[1])

lowest = firsthalfLowHigh[0]

else

lowest = secondhalfLowHigh[0]

if(firsthalfLowHigh[1] < secondhalfLowHigh[1])

highest = firsthalfLowHigh[1]

else

highest = secondhalfLowHigh[1]

return (lowest,highest)

this pseudo code divides the unsorted list in two until one value is in a list and passes each one intp the function again, it then returns that list and compares that to the other values returned from the other call, comparing its lowest and highest value found.