1.)

A screen shot of a computer program

Description automatically generated with low confidence

2.)

a.) A Python Dictionary is an Array object.

b.) dic.update({key,value})

c.) dic[“key”] = “val2”

d.) The for loop will be executed for however many key:value pairs are in the dictionary

3.)

A screen shot of a computer program

Description automatically generated with low confidence

4.)

a.) A picture containing text, screenshot, software, multimedia software

Description automatically generated

b.) C1:

i = 0 and ans = [20]

i = 1 and ans = [20]

i = 2 and ans = [20, 70]

i = 3 and ans = [20, 70, 30]

i = 4 and ans = [20, 70, 30, 10]

i = 5 and ans = [20, 70, 30, 10, 80]

i = 6 and ans = [20, 70, 30, 10, 80]

i = 7 and ans = [20, 70, 30, 10, 80, 90]

i = 8 and ans = [20, 70, 30, 10, 80, 90]

C2:

i = 0 and ans = [35]

i = 1 and ans = [35, 45]

i = 2 and ans = [35, 45, 55]

i = 3 and ans = [35, 45, 55]

i = 4 and ans = [35, 45, 55]

i = 5 and ans = [35, 45, 55]

C1 + C2:

[20, 70, 30, 10, 80, 90, 35, 45, 55]

c.)

180

181

182 c1

183 c2 \* n-1

184 c3 \* n-1

185 c4 \* (m-1)\*m/2

186 c5 \* (m-1)\*m/2

187 c6 \* (m-1)\*m/2

188 c7 \*n-1

189 c8 \* n-1

190 c9

191

192 c10

193 c11

193 c12

A piece of paper with math equations

Description automatically generated with low confidence

5.)

A piece of paper with writing on it

Description automatically generated with medium confidence

6.)

A paper with writing on it

Description automatically generated with low confidence

A picture containing text, drawing, sketch, paper

Description automatically generated

7.)

a.)

A picture containing text, screenshot, software, font

Description automatically generated

b.)

Params: [17, 62, 49, 73, 26, 51] 5

Params: [17, 62, 49, 73, 26, 51] 4

Params: [17, 62, 49, 73, 26, 51] 3

Params: [17, 62, 49, 73, 26, 51] 2

Params: [17, 62, 49, 73, 26, 51] 1

Params: [17, 62, 49, 73, 26, 51] 0

Return: 17

62 compared to 17

Return: 62

49 compared to 62

Return: 62

73 compared to 62

Return: 73

26 compared to 73

Return: 73

51 compared to 73

Return: 73

A piece of paper with writing on it

Description automatically generated with medium confidence

8.)

a.) MergeSort is Big-Oh class n\*lg(n) because the fundamental step of the algorithm is the comparisons in the Merge function. In the worst case the cost of Merge is n-1 and Merge will be called however many times it takes to combine all subarrays into one final array by combining two at a time. QuickSort is in Big-Oh class n^2 because in the worst case when the pivot is always on at the left or right index, the cost of the partition function can be n-1(n)/2.

b.) MergeSort average case is n\*lg(n) because no algorithm will do more or less than that amount of work. QuickSort average case is n\*lg(n) because the best case is n\*lg(n) when the partition algorithm does n amount of work. The worst case for QuickSort is very unlikely, so the average case is best estimated using the best case.

c.) No answer. Ran out of time ☹

d.) No answer. Ran out of time ☹