2383 Assignment

# a,b,c,d,e,f,g,h,i,j,k,l

Diagram

Description automatically generated

Then

A picture containing indoor

Description automatically generated

Then

Diagram

Description automatically generated

Then

A picture containing nature

Description automatically generated

Then

Diagram

Description automatically generated

Finally

Diagram

Description automatically generated

# a,v,l,t,r,e,i,s,f,u,n

Diagram, shape

Description automatically generated

Then

Diagram

Description automatically generated

Diagram

Description automatically generated with medium confidence

Shape

Description automatically generated

Shape

Description automatically generated

A picture containing diagram

Description automatically generated

Diagram

Description automatically generated with medium confidence

Shape

Description automatically generated with medium confidence

Diagram

Description automatically generated with medium confidenceDiagram

Description automatically generated with medium confidence

Diagram

Description automatically generated

Algorithm: findAllInRange (n, k1, k2)

Input: Node n, int k1, int k2

findAllInRange(n,k1,k2){

count ← 0;

if(n = null){

return;

}

if (k1< n.data)

{

findAllInRange(n.left, k1, k2);

}

if (k1 <= n.data AND k2 >= n.data) {

print(n.data + " ");

count++;

}

if (k2 > n.data) {

findAllInRange(n.right, k1, k2);

}

}

findAllInRange(int k1, int k2)

{

findAllInRange(root, k1, k2);

}

Time complexity is O(n).­­­­­

# 3-

Table

Description automatically generated

Graphical user interface, table, Excel

Description automatically generated

# 4-

A piece of paper with writing

Description automatically generated with medium confidence

Text, letter

Description automatically generated

A piece of paper with writing

Description automatically generated with medium confidence

Text, letter

Description automatically generated

Text, letter

Description automatically generated

# 5-

Algorithm: exchange(array, ind1, ind2)

Input: int[] array, int ind1, ind2

exchange(array, ind1, ind2){

temp ← array[ind1]

array[ind1] ← array[ind2]

array[ind2] ← temp;

}

Algorithm: gRI(min, max)

Input: int min, int max

gRI(min, max){

r ← new Random()

lim ← (max-min)+1

rni ← r.nextInt(lim)+min

return rni

}

Algorithm: fKL(array, k)

Input: int[] array, int k

fKL(array, k){

if array.length = 1 then

return array[0]

return sR(array, 0, array.length-1, k)

}

Algorithm: sR(array, start, end, k)

Input: int[] array, inst start, int end, int k

sR(array, start, end, k){

ind ← rP(array, start, end)

if (ind = (k-1))

return array[ind]

else if(ind >= k)

return sR(array, start, ind-1, k)

else

return sR(array, ind+1, end, k)

}

Algorithm: rP(array, start, end)

Input: int[] array, int start, int end

rP(array, start, end){

rI ← gRi (start, end)

exchange(array, rI)

piv ← array[end]

p ← start-1

for (i ← start to end){

if (array[i]>=piv){

p++

exchange(array, i, p)

}

}

exchange(array, p+1, end)

return p++

}

Time complexity is O(n)