Question 1 :

a- (2^(n+1)-1) = 23 so n approx 4 levels - log(n)

b- quickly

c- i will assume regions equivalent to levels (2^(R+1)-1) = n where R is the no of regions

d- better

e- n(n-1) = 8\*7 = 56

f- inorder

g- all but more logical is postorder

h- open

i- array

------------------------------------------------

Question 2 :

a- False - instead of max heap put BST

b- True

c- False - at the most left node

d- False - faster

e- True

f- False - instead of inorder put postorder

-----------------------------------------------

Question 3 :

a- (5+3)/2 \* (2+8 - (3\*2)) = 4 \* 4 = 16

b- inorder : 5 + 3 / 2 \* 2 + 8 - 3 \* 2 /\*\*\*/ postorder : 5 3 + 2 / 2 8 + 3 2 \* - \*

c- #pragma once

class TreeNode

{

char value;

TreeNode \*Right;

TreeNode \*Left;

public:

TreeNode(char v);

TreeNode\* GetRight();

TreeNode\* GetLeft();

void SetRight(TreeNode \*R);

void SetLeft(TreeNode \*L);

char GetValue();

void SetValue(int v);

};

#include "TreeNode.h"

TreeNode::TreeNode(char v)

{

value = v;

Right = nullptr;

Left = nullptr;

}

TreeNode\* TreeNode::GetRight()

{

return Right;

}

TreeNode\* TreeNode::GetLeft()

{

return Left;

}

void TreeNode::SetRight(TreeNode \*R)

{

Right = R;

}

void TreeNode::SetLeft(TreeNode \*L)

{

Left = L;

}

char TreeNode::GetValue()

{

return value;

}

void TreeNode::SetValue(char v)

{

value = v;

}

d- void prefix\_exp (TreeNode\* root)

{

if(root == NULL)

return ;

cout<<root->GetValue()<<" ";

prefix\_exp(root->GetLeft());

prefix\_exp(root->GetRight());

}

e- void Tree::no\_leaf\_nodes(TreeNode \*root,int& count)

{

if (root == NULL)

{

return ;

}

no\_leaf\_nodes(root->GetLeft(),count);

no\_leaf\_nodes(root->GetRight(),count);

if (root->GetLeft() == NULL && root->GetRight() == NULL)

count++;

}

you need just only to send root and count initialized to 0

--------------------------------------------------

Question 4 :

a- arr[] = {100,19,36,17,3,25,1,2,7}

b- arr[2n+2] = arr[2\*3+2] = arr[8] = 7

c- 100

21 36

17 19 25 1

2 7 3

arr[] = {100,21,36,17,19,25,1,2,7,3}

d- 36

19 25

17 3 7 1

2

e- o(1) if element isn't removed from the heap otherwise o(log(n))

-----------------------------------------------------

Question 6 :

a- (1+2+1+2+3+1+2+3+1+2+3)/11 =1.909...etc

b- index 0 0->49

index 1 1->36->64->8->29

index 2 2->16->100->49

index 3 10

index 4 4->25->81->18

index 5

index 6 27

c- no , i think (k%7) is better than (k\*k)%7 as it is stretching keys on buckets and decreases no of collisions

d- first sorting 0 , 1 , 2 , 4 , 16 , 25 , 36 , 49 , 64 , 81 , 100

index 0 0

index 1 1

index 2 2

index 3 81

index 4 4

index 5 25

index 6 64

index 7 100

index 8

index 9 49

index 10

index 11

index 12

index 13

index 14

index 15

index 16 16

index 17 36

index 18

index 19

no of collisions = 1 + 2 + 2 + 7 = 12 collisions