CS & IT ENGINEERING

Data Structure

Tree – 4

DPP 04 **Discussion Notes**

By- Pankaj Sharma sir

01 Question

02 Discussion

Q.1

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
  };
  int func(struct treenode *p, struct treenode *q){
  if(p==NULL && q==NULL) return 1;
  if((!p && q) || (!q && p)) return 0;
  return (p->data==q->data) && func(p->left, q->right) && func(p->right, q-
  >left);
  }
```

Initially the addresses of root node of two trees are passed into p and q respectively, the function-

A. Returns 1 iff the two trees are identical.

B. Returns 1 iff the two trees are mirror images of each other.

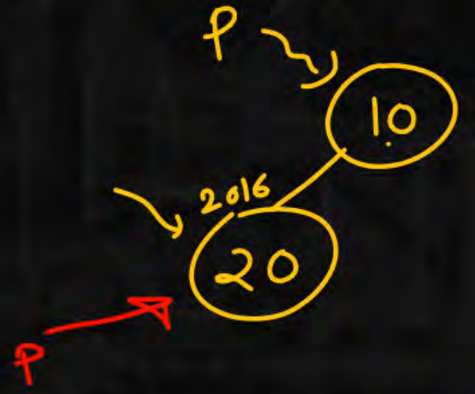
C. Returns 1 iff the two trees emerge from the same root node.

D. None of the above.

Q.1

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
 };
int func(struct treenode *p, struct treenode *q){
if(p==NULL && q==NULL) return 1;
if((!p && q) || (!q && p)) return 0;
return (p->data==q->data) && func(p->left, q->right) && func(p->right, q-
>left);
}
```



1 && func(2016, 1012)

&& func(NULL, NULL)

func(NULL, NULL) && func(NULL, NULL)

Initially the addresses of root node of two trees are passed into p and q respectively, the function-

A. Returns 1 iff the two trees are identical.

B. Returns 1 iff the two trees are mirror images of each other.

C. Returns 1 iff the two trees emerge from the same root node.

D. None of the above.

Q.1

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
  };
  int func(struct treenode *p, struct treenode *q){
  if(p==NULL && q==NULL) return 1;
  if((!p && q) || (!q && p)) return 0;
  return (p->data==q->data) && func(p->left, q->right) && func(p->right, q-
  >left);
  }
```

B

Initially the addresses of root node of two trees are passed into p and q respectively, the function-

A. Returns 1 iff the two trees are identical.

B. Returns 1 iff the two trees are mirror images of each other.

C. Returns 1 iff the two trees emerge from the same root node.

D. None of the above.

Q.2

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
    };
    int func(struct treenode *p, struct treenode *q){
    if(p==NULL && q==NULL) return 1;
    if((!p && q) || (!q && p)) return 0;
    return (p->data==q->data) && func(p->left, q->left) && func( p->right, q >right);
    }
```

Initially the addresses of root node of two trees are passed into p and q respectively, the function-

A. Returns 1 iff the two trees are identical.

B. Returns 1 iff the two trees are mirror images of each other.

C. Returns 1 iff the two trees emerge from the same root node.

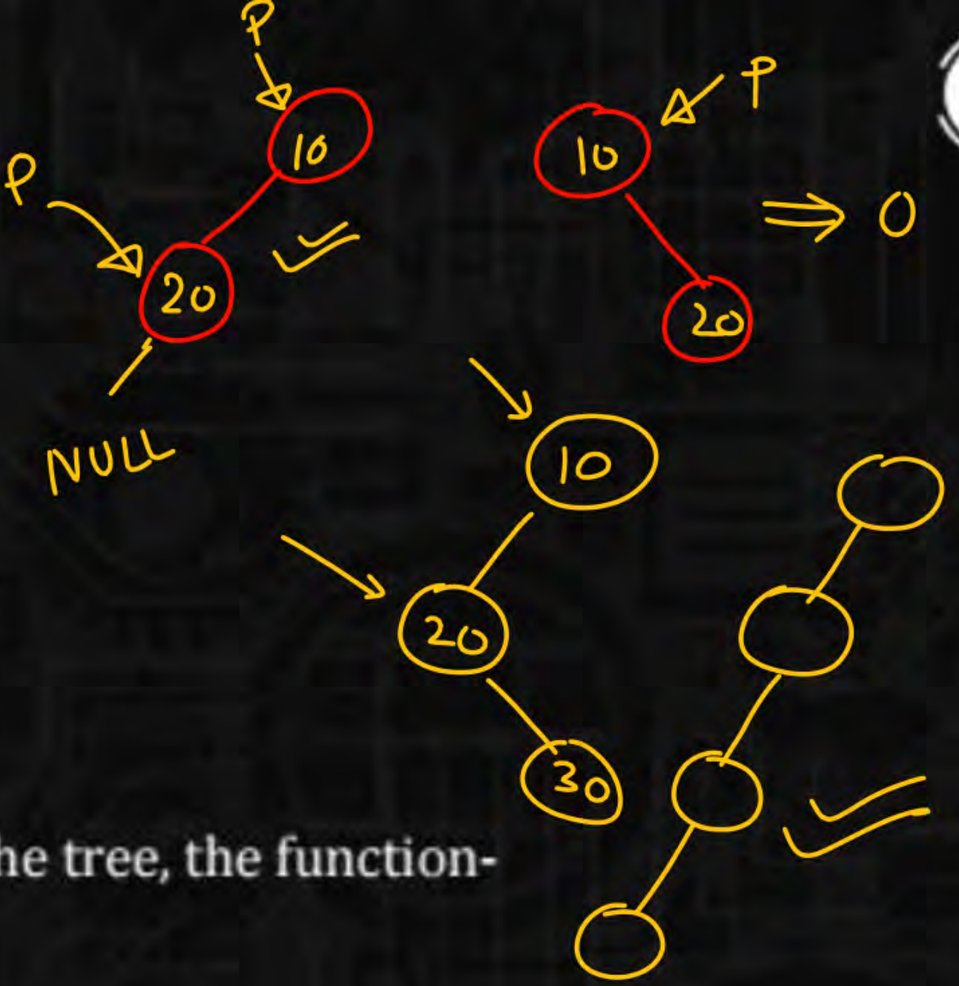
D. None of the above

Q.3

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
    };
  int func(struct treenode *p){
      if(p==NULL) return 1;
      else if(p->right!=NULL) return 0;
      return func(p->left);
      }
```

Initially p contains the root node address of the tree, the function-



A. Returns 1 if a binary tree is left-skewed.

B. Returns 1 if a binary tree is right-skewed.

C. Returns 1 if a binary tree is not right-skewed.

D. None of the above.

Q.4

Consider the following functions:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
 };
int f1(struct treenode *t){
     if(t==NULL) return 1;
     else if(t->left!=NULL) return 0;
     return func(t->right);
}
int * f2 (struct treenode *t){
if(t==NULL) return 1;
else if(t->left==NULL && t->right==NULL)
return 1;
else if
((t → left → data < t->data) && (t → right → data > t->data))
return func(t->left) && func(t->right);
else
return 0;
}
 int f3(){return f2(t) && f1(t);}
```

Right Skewed

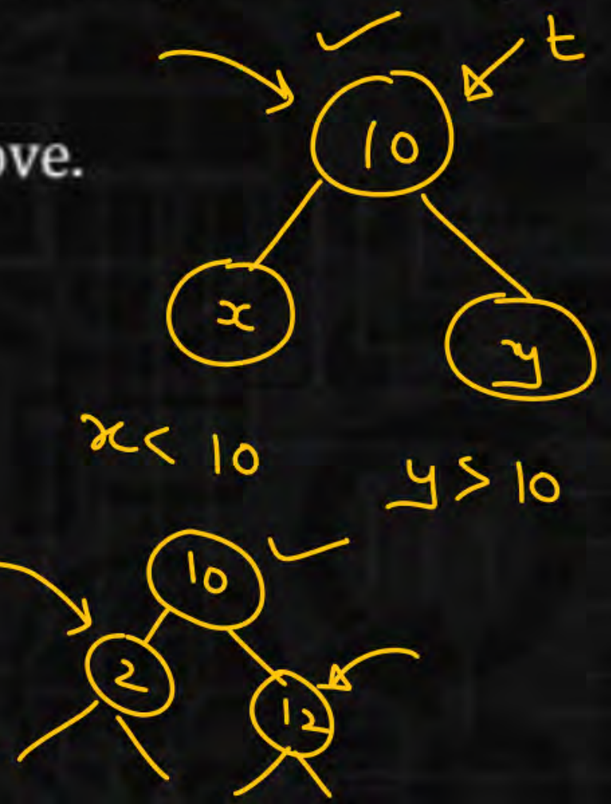
Assume, t is a pointer to the root node of a binary tree, the function f(3):

A. Returns 1 if the binary tree is a left skewed BST

B. Returns 1 if the binary tree is not a left skewed BST

C. Returns 1 if the binary tree is a right skewed BST

D. None of the above.



Q.4

Consider the following functions:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
 };
int f1(struct treenode *t){
     if(t==NULL) return 1;
     else if(t->left!=NULL) return 0;
     return func(t->right);
}
int * f2 (struct treenode *t){
if(t==NULL) return 1;
else if(t->left==NULL && t->right==NULL)
return 1;
else if
((t → left → data < t->data) && (t → right → data > t->data))
return func(t->left) && func(t->right);
else
return 0;
}
 int f3(){return f2(t) && f1(t);}
```

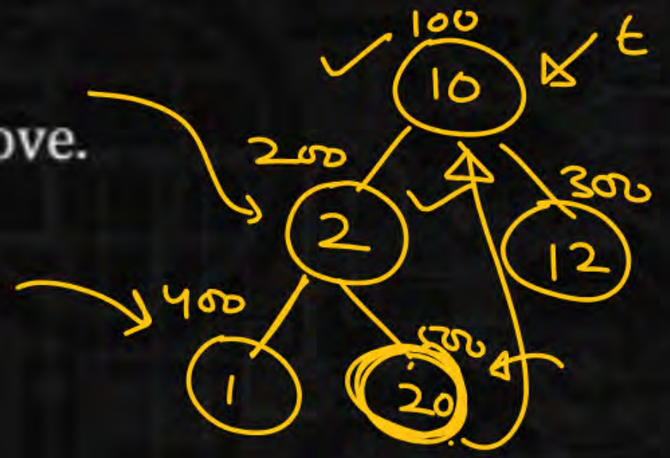
Assume, t is a pointer to the root node of a binary tree, the function f(3):

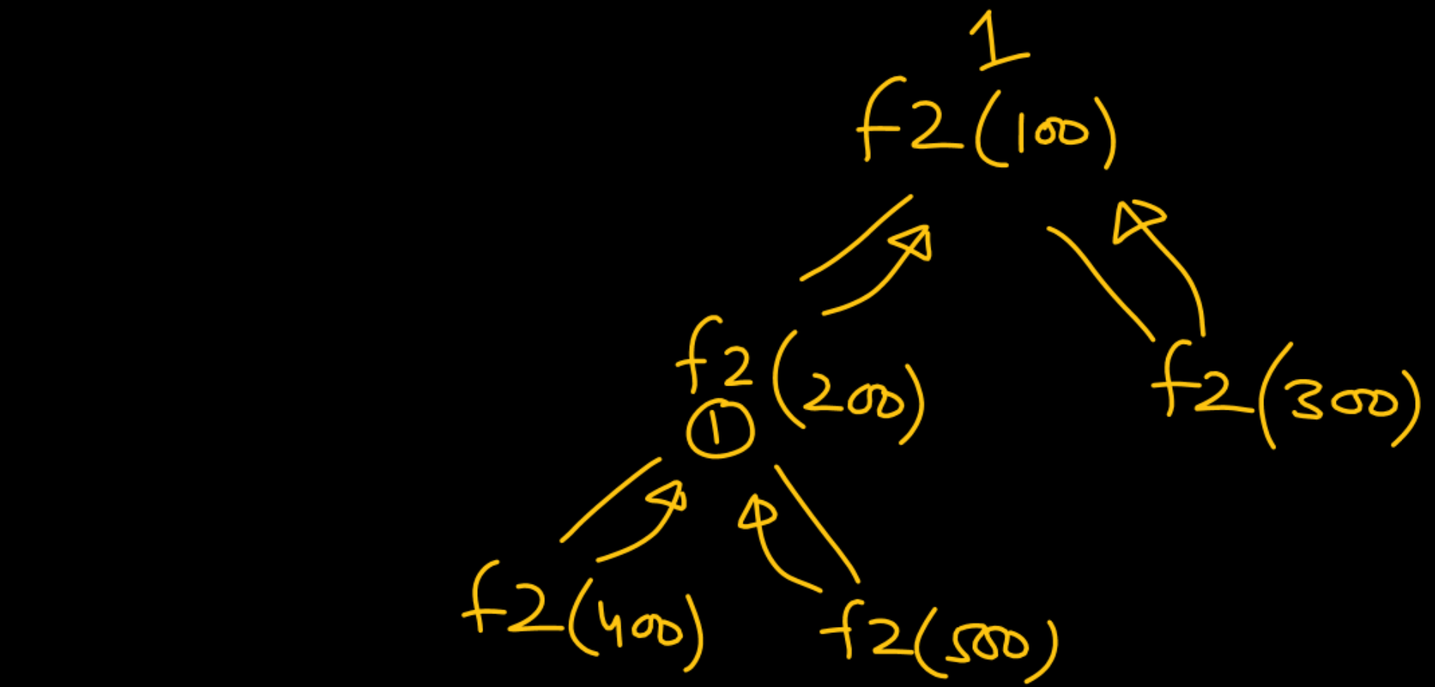
A. Returns 1 if the binary tree is a left skewed BST

B. Returns 1 if the binary tree is not a left skewed BST

C. Returns 1 if the binary tree is a right skewed BST

D. None of the above.

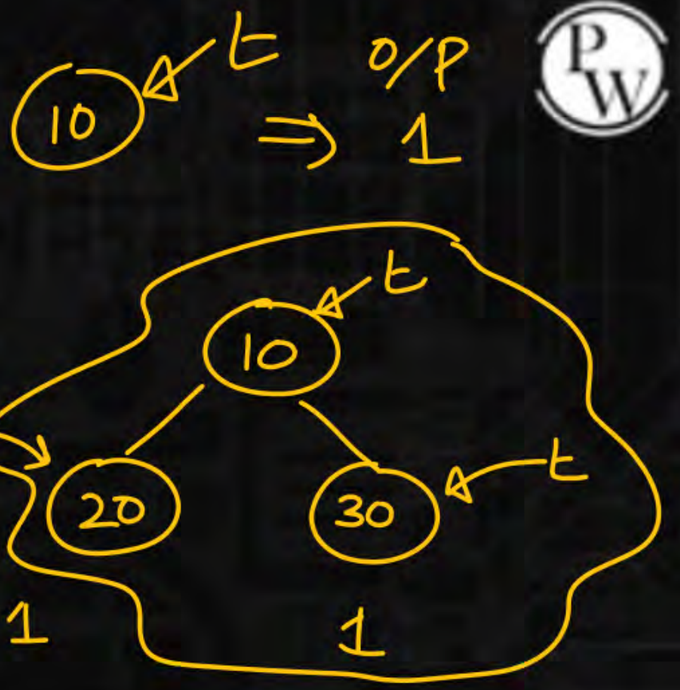



1
f2(100)
f2(200)
1
f2(300)
f2(400)
f2(500)

Q.5

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
 };
int func(struct treenode *t){
        if(t==NULL) return 0;
elseif(t->left==NULL && t->right==NULL) return 1;
        else
        return 1+func(t->left)+func(t->right);
}
```



Assume, t is a pointer to the root node of a binary tree, the function computes-

A. Number of leaf nodes in the binary tree

B. Number of internal nodes in the binary tree

C. Total number of nodes in the binary tree

D. None of the above

Q.6

The given tree is passed to the following function:

```
void func(struct treenode *t)
{
  if(t)
    {
1   printf("%d", t->data);
2   func(t->right);
3   printf("%d", t->data);
4   func(t->left);
    }
}
```

The output string is-

AEFFEABDDBCC

A. AEFFEBDDCCBA

B. AEFFEABDDBCC

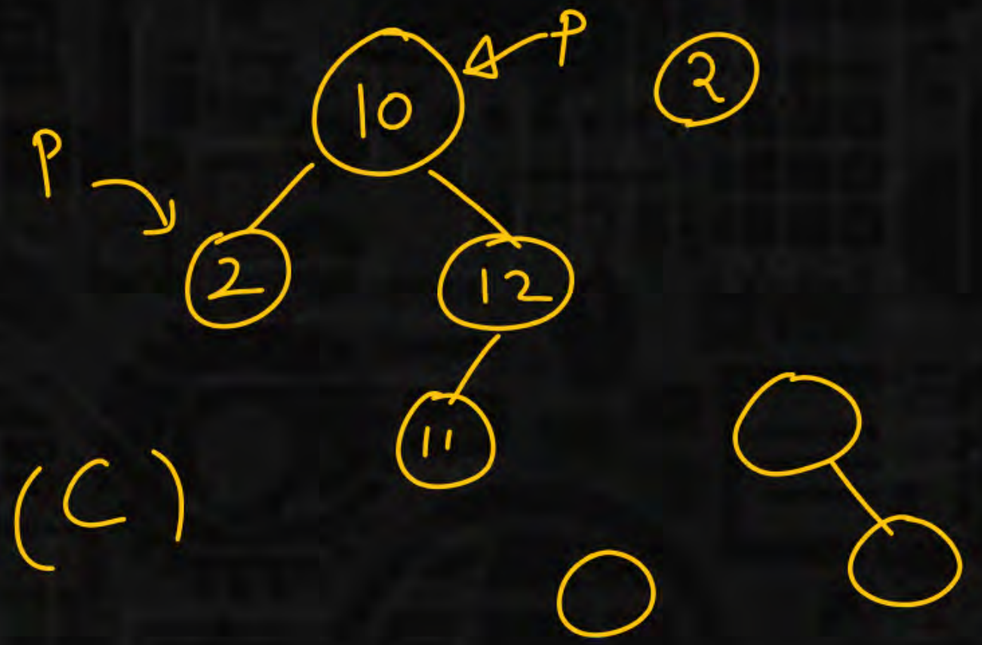
C. AEFFEBDDCCBA

D. None of the above

Q.7

Consider the following function:

```
struct treenode{
        struct treenode *left;
        int data;
        struct treenode *right;
  };
  void func(struct treenode *p){
  while(p->left!=NULL) p=p->left;
  printf("%d", p->data);
}
```



If the address of the root node of the BST is passed to p, the above function prints-
(Assume, the tree contains at least one node)

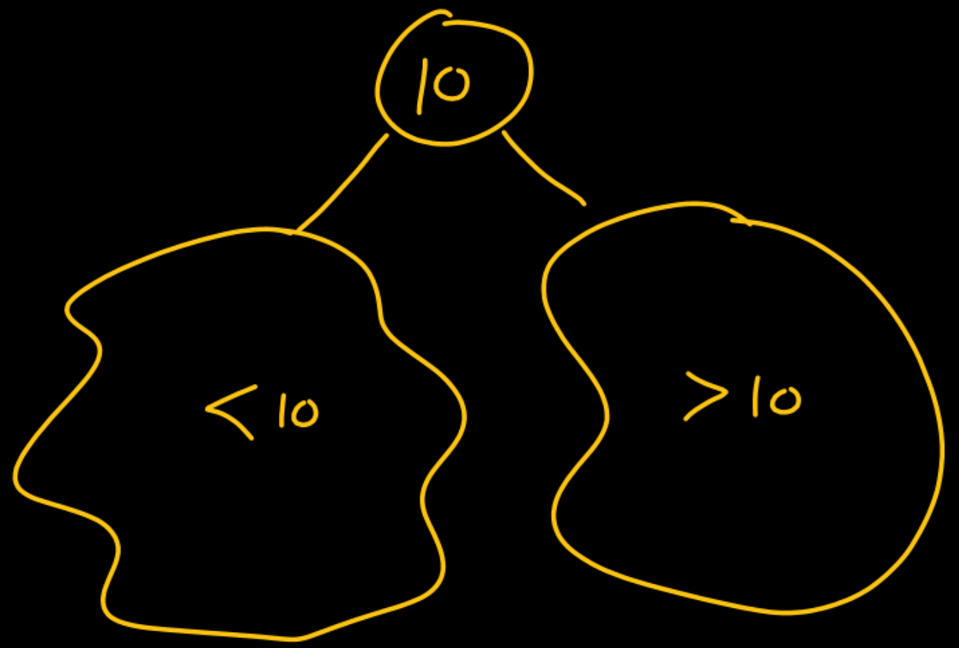
A. The maximum element in the BST

B. The ancestor of two leftmost leaf nodes

C. The minimum element in BST

D. None of the above


10
5
greater than 5


10
< 10
> 10

Q.8

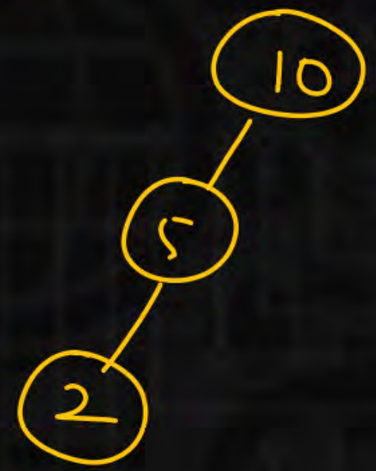
Consider the following two statements:

P: The minimum number of nodes in a complete binary tree of height h is 2^{h+1}.

$2^h \leq n \leq 2^{h+1}$

Q: A binary search tree is always a complete binary tree.

Which of the statement(s) is/are CORRECT?



(D)

A. P only

B. Q only

C. Both P and Q

D. Neither P nor Q


THANK YOU GW
SOLDIERS !