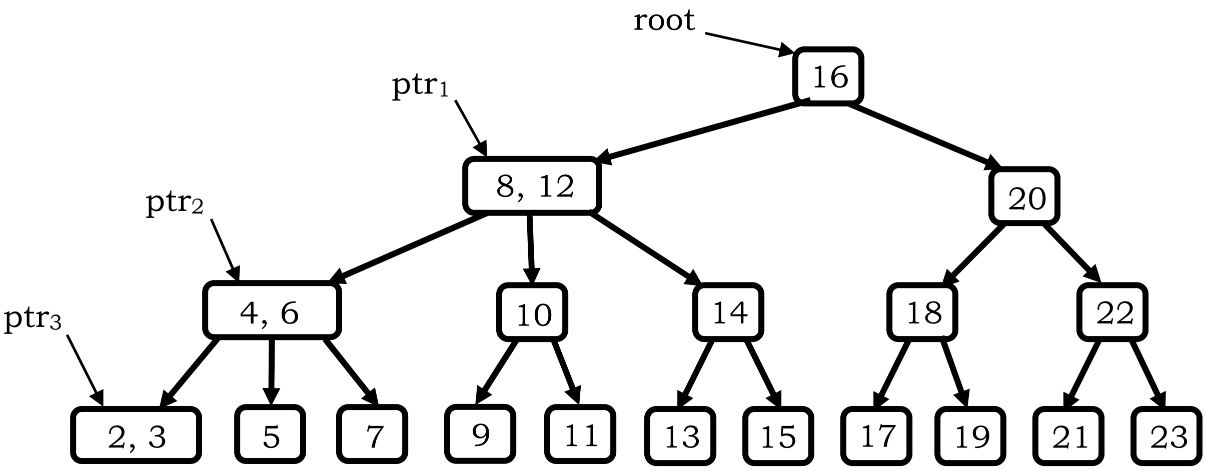
2-3-tree insertion

```
     template <typename T> struct node
     { T d1, d2;
       node * L, * M, * R;
       int num; //          if num is 1, d2 and M are not used.

       node(node * nL, T nD, node * nR): L(nL), d1(nD), R(nR), num(1) { } };

bool insert(node * ptr, T value, node * & upL, T & upD, node * & upR)
{ if (ptr == NULL)
  { upL = NULL;
    upD = value;
    upR = NULL;
    return true; }

  else if (value < ptr->d1)
  { bool split = insert(ptr->L, value, upL, upD, upR);
    if (! split)
      return false; // not split means insertion process is complete
                    // split means immediate child broke apart, fragments in last 3
                    // parameters must be reincorporated into the tree.

    if (ptr->num == 1)
    { ptr->d2 = ptr->d1;
      ptr->L = upL;
      ptr->d1 = upD;
      ptr->M = upR;
      ptr->num = 2;
      return false; }

    else
    { upL = new node(upL, upD, upR);
      upD = ptr->d1;
      ptr->d1 = ptr->d2;
      ptr->L = ptr->M;
      ptr->num = 1;
      upR = ptr;
      return true; } }

  else if (ptr->num == 1 || value > ptr->d2)
  { bool split = insert(ptr->R, value, upL, upD, upR);
    if (! split)
      return false;
    /* similar to insertion in left */ }

  else
  { bool split = insert(ptr->M, value, upL, upD, upR);
    if (! split)
      return false;
    /* similar to insertion in left */ } }
```

```
void insert(node * & root, T value)
{ node * fL, * fR;
  T fD;
  bool spilt = insert(root, value, fL, fD, fR);
  if (split)
    root = new node(fL, rD, fR); }
```



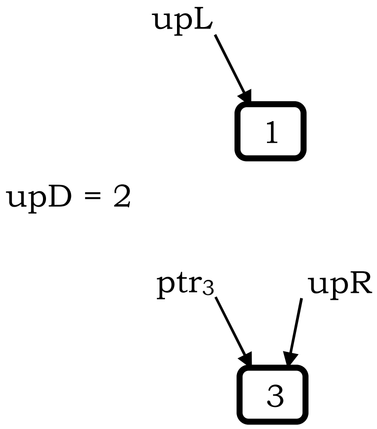
call 1: insert(root, 1);

causes call 2: insert(ptr_1, 1)

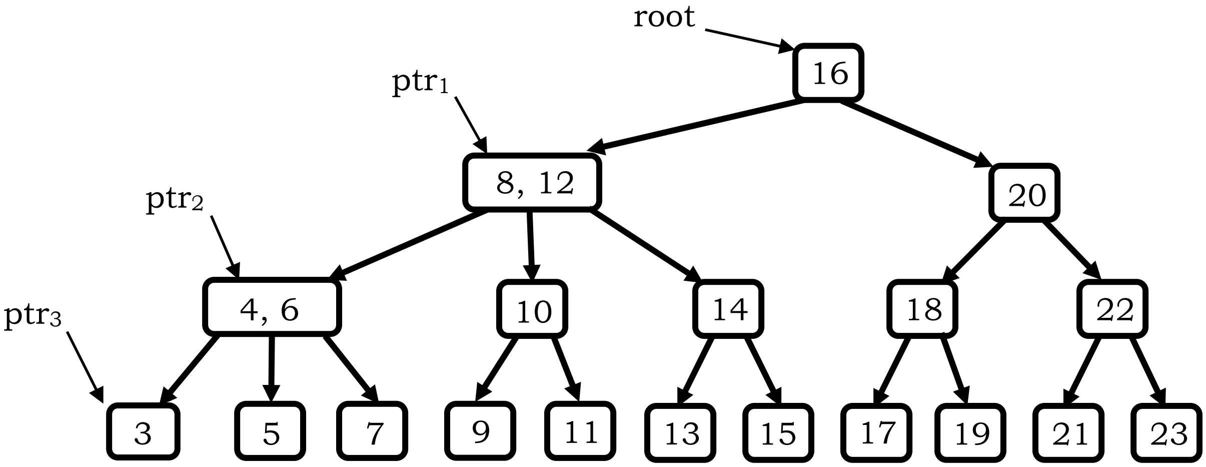
causes call 3: insert(ptr_2, 1)

causes call 4: insert(ptr_3, 1)

causes call 5: insert(NULL, 1)
upL = NULL, upD = 1, upR = NULL
return true



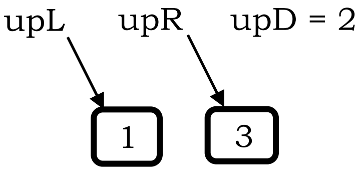
upL = new node(upL, upD, upR)
= new node(NULL, 1, NULL)
upD = 2
ptr_3 ->d1 = ptr_3 ->d2
= 3
ptr_3 ->L = ptr_3 ->M
= NULL
ptr_3 ->num = 1
upR = ptr_3
return true



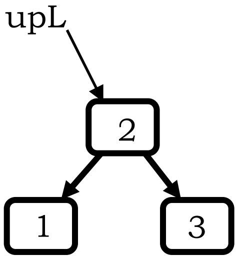
call 1: insert(root, 1);

caused call 2: insert(ptr$_1$, 1)

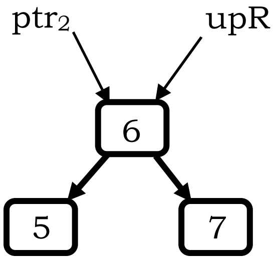
caused call 3: insert(ptr$_2$, 1)

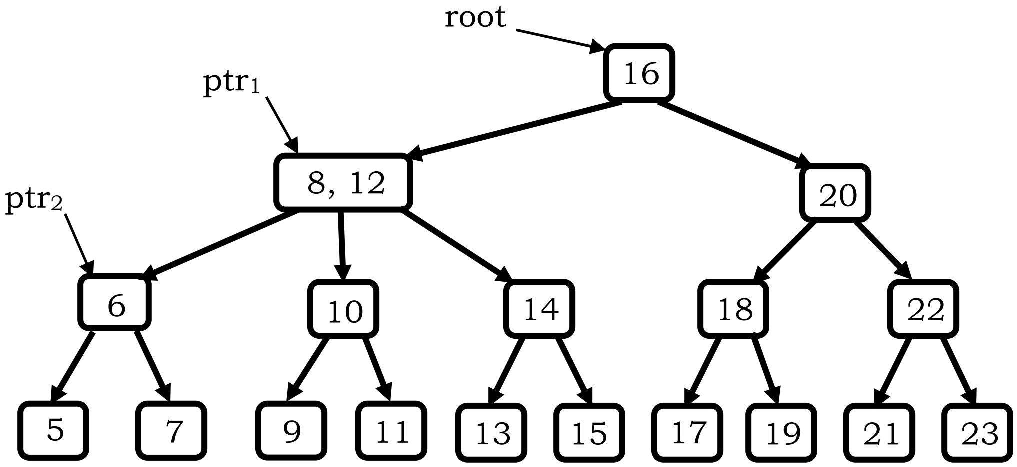


caused call 4: (all done)



```
upL = new node(upL, upD, upR)
upD = ptr$_2$ ->d1
        = 4
ptr$_2$ ->d1 = ptr$_2$ ->d2
               = 6
ptr$_2$ ->L = ptr$_2$ ->M
ptr$_2$ ->num = 1
upR = ptr$_2$
return true
```

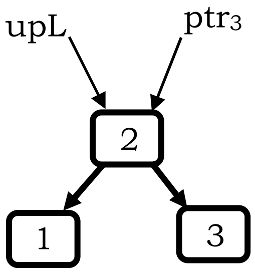




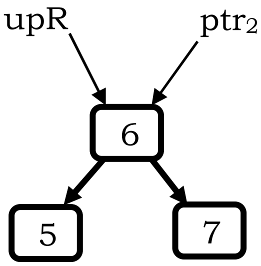
call 1: insert(root, 1);

caused call 2: insert(ptr_1, 1)

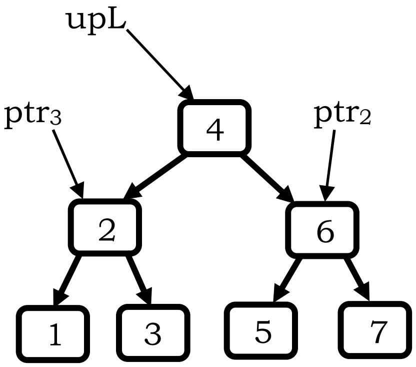
caused call 3: (all done)

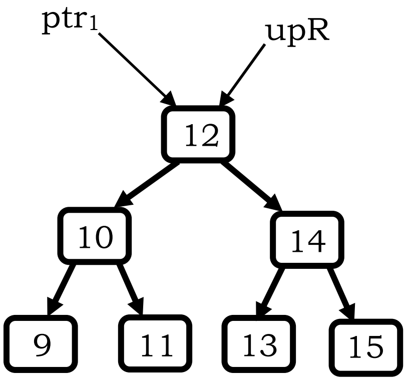


upD = 4

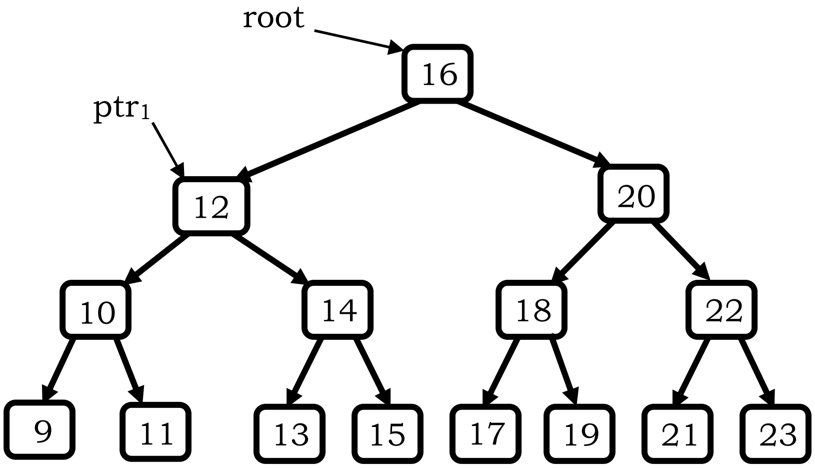


upL = new node(upL, upD, upR)
upD = ptr_1 ->d1
= 8
ptr_1 ->d1 = ptr_1 ->d2
= 12
ptr_1 ->L = ptr_1 ->M
ptr_1 ->num = 1
upR = ptr_1
return true



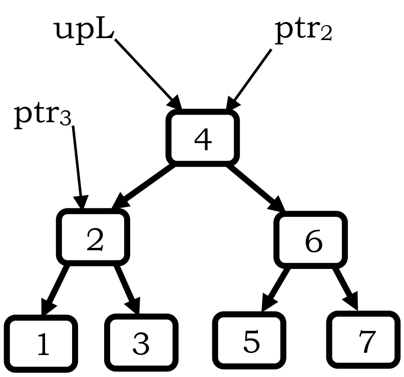


upD = 8

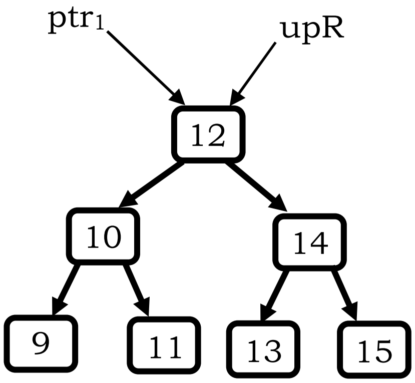


call 1: insert(root, 1);

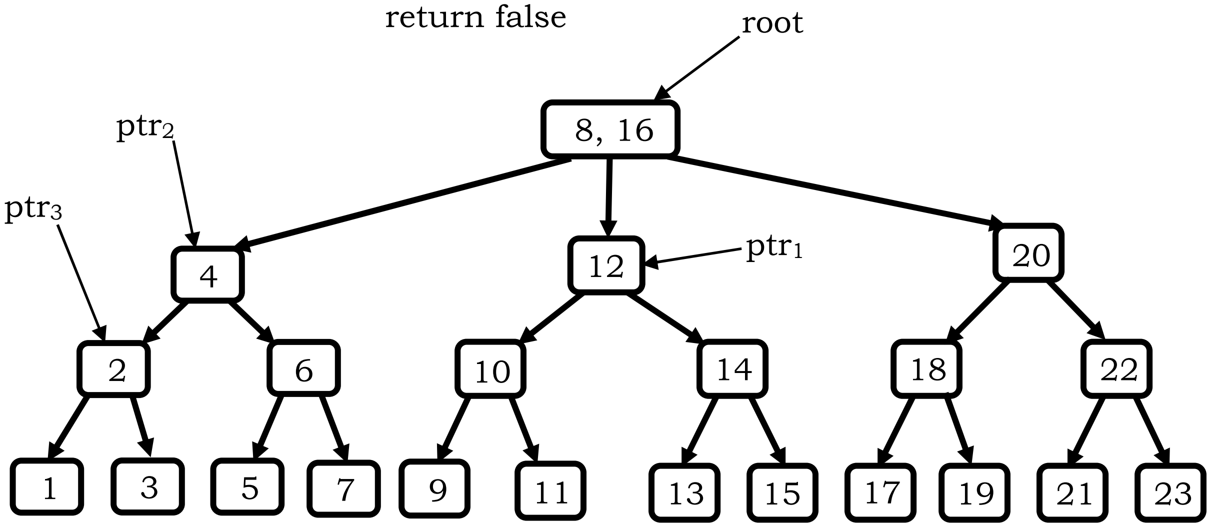
causes call 2: (all done)



upD = 8



```
root ->d2 = root ->d1
root ->L = upL
root ->d1 = upD
root ->M = upR
root ->num = 2
return false
```


root
8, 16
ptr$_2$
ptr$_3$
4
12
ptr$_1$
20
2
6
10
14
18
22
1
3
5
7
9
11
13
15
17
19
21
23