Binary Search Tree:

For every parent node X,

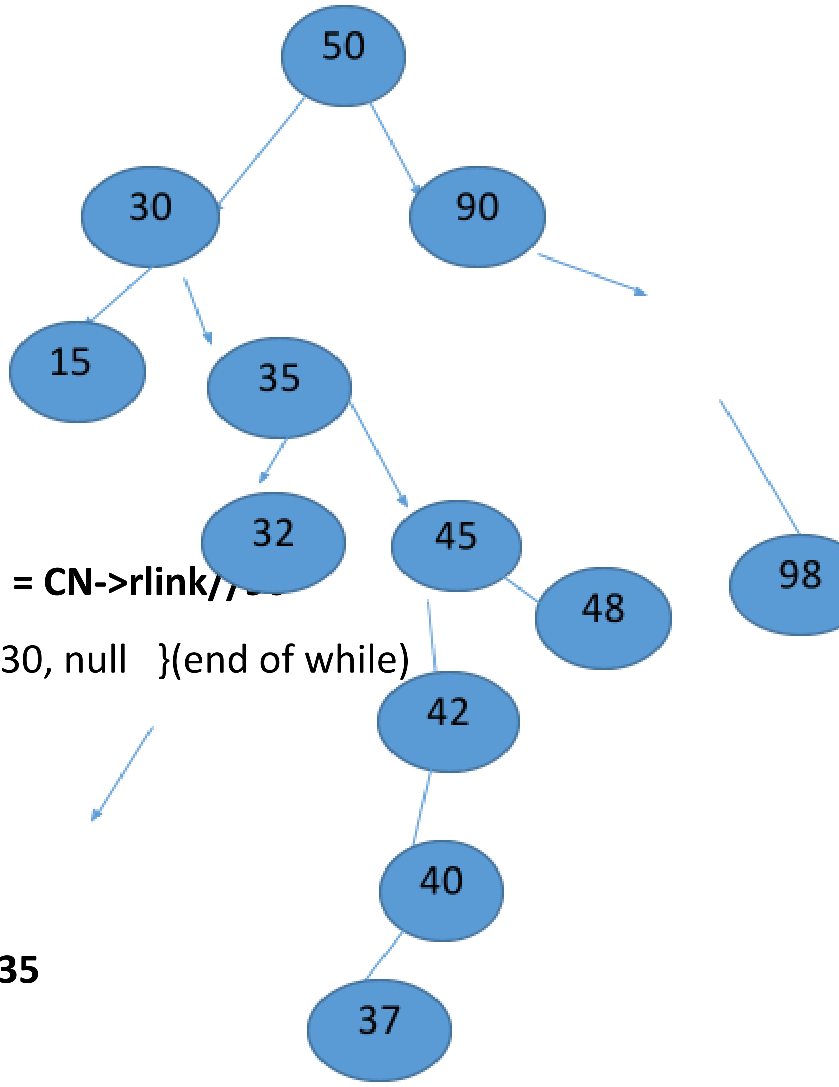
Key(X) > Key(Leftchild) and

Key(X) < Key(Rightchild)

Implement Dictionary-> Insert, Search , Update, Delete

Applications: Searching, Sorting, Dictionary implementation, Priority queue implementation.

```
NN=30, 90 15 ,95 35

CN=R

PN = NULL

       CN!=NULL)

    {

         PN = CN // 50 ,90

        if(NN->info>CN->info) CN = CN->rlink//

 else     CN=CN->llink  //  null  , 30, null   }(end of while)

        if(PN->info>NN->info)

           PN->llink=NN; //30

        Else

        PN->rlink = NN;  90 , 95 , 35
```



15 30 32 35 37 40 42 45 48 50 90 95 98

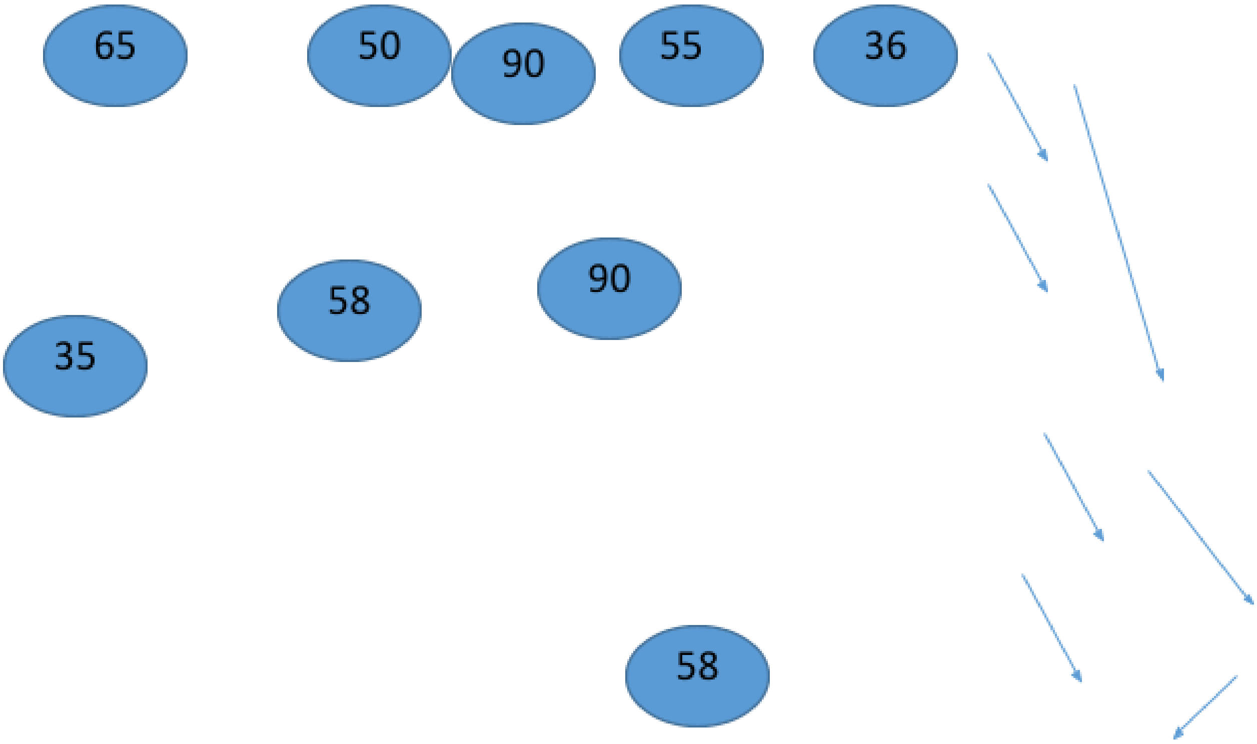
50 30 15 35 32 45 42 40 37 48 90 95 98

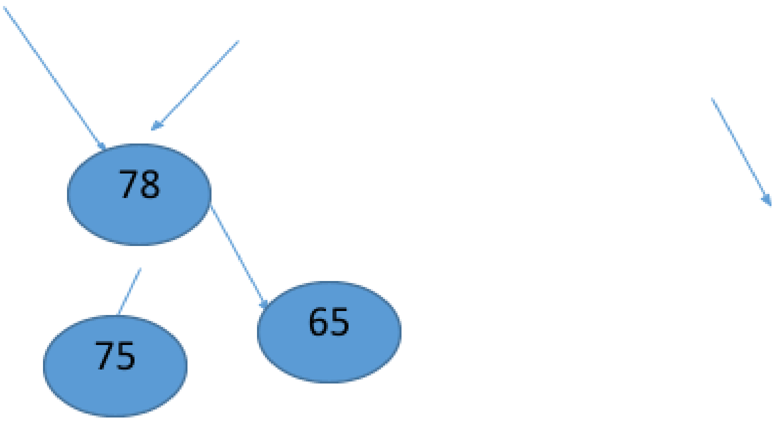
15 32 37 40 42 48 45 35 30 98 95 90 50

15 30 35 36 37 40 42 45 48 50 90 95 98

50 30 15 40 35 37 36 45 42 48 90 95 98

65 50 90 55 36 35 58 90 75 65 78 58



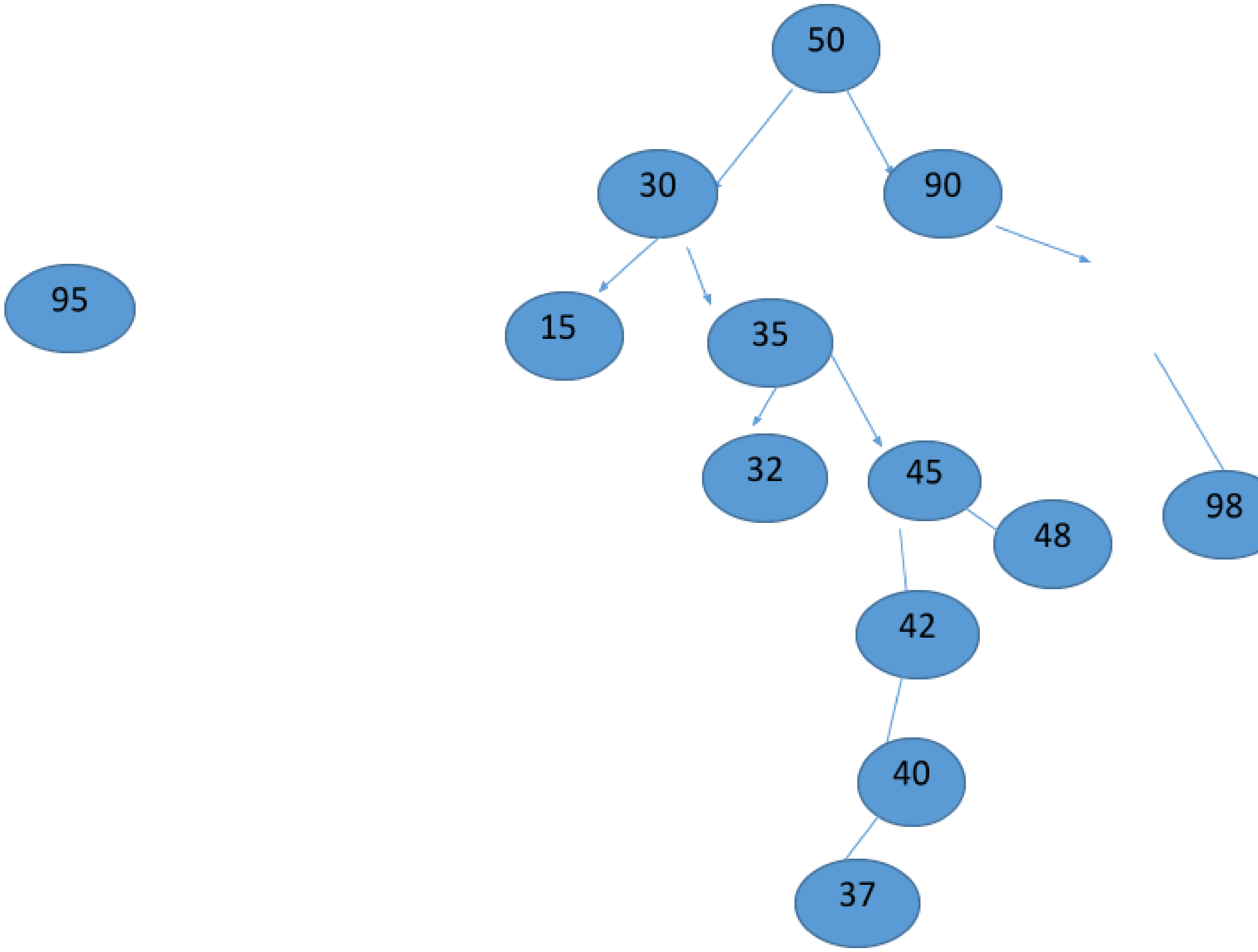


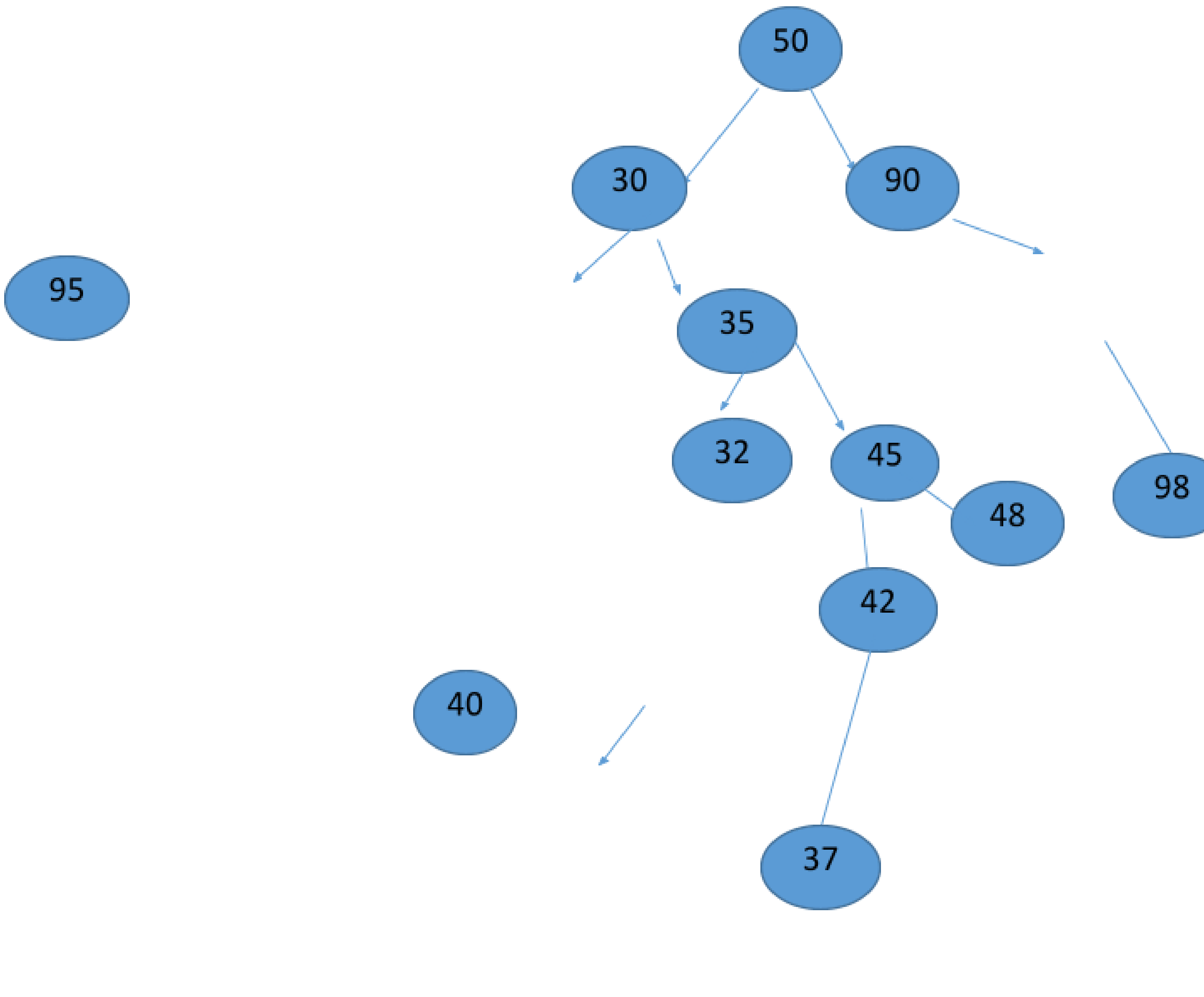
R = 58

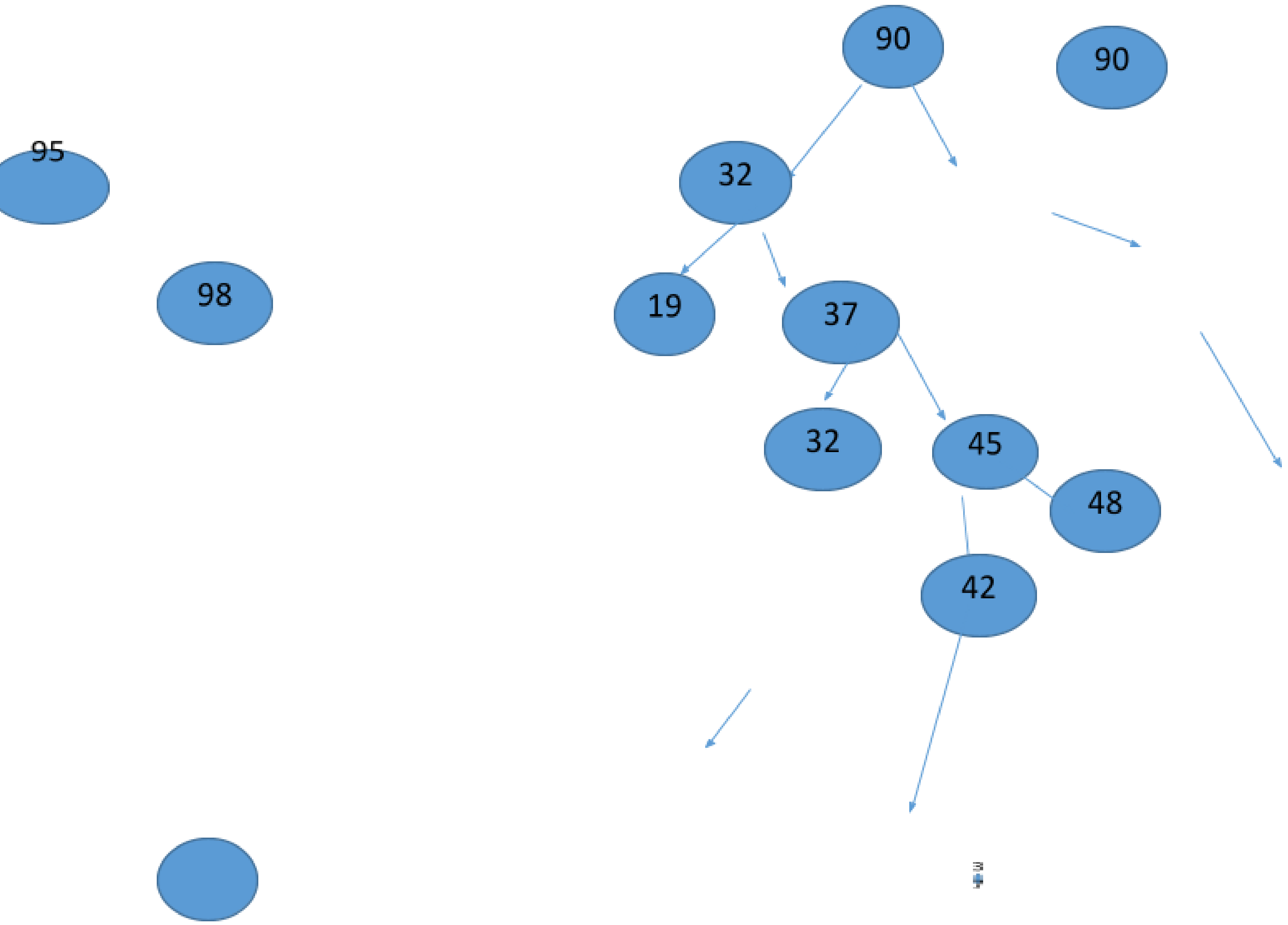
NS = 58 PN= NULL

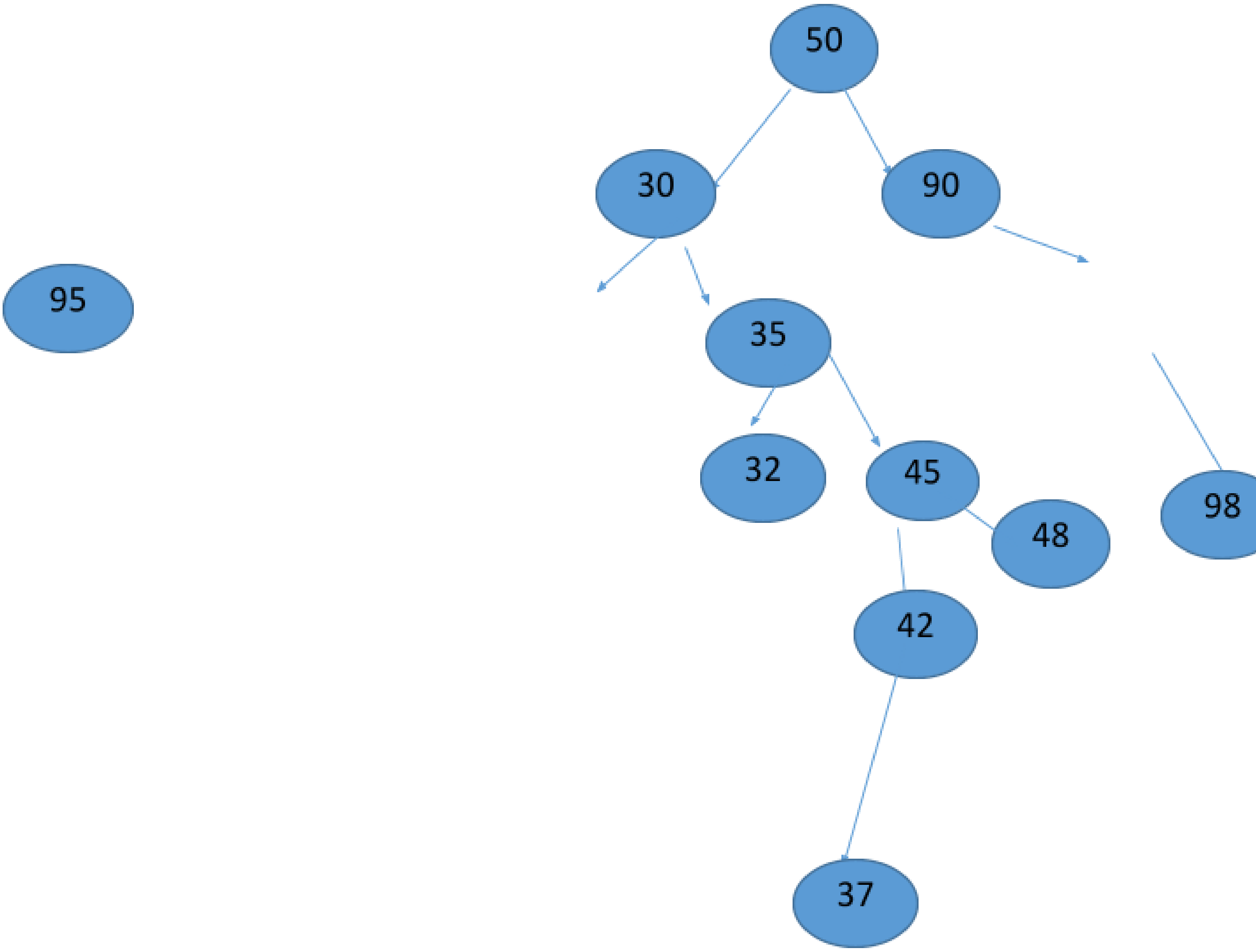
```
IF(PN==NULL)
  R= NS->RLINK
R =ns->LLINK
```

Delete: 15 40 35 50




50
30
90
95
35
32
45
48
98
42
40
37


90
90
95
32
98
19
37
32
45
48
42


50
30
90
95
35
32
45
48
98
42
37

```
Insert(Root)
{
   // Create NN
// Read info and assign left and right links to null
If Root  == Null return NN
PN = NULL
CN = Root;
While(CN !=NULL)
{
   PN= CN;
  If(NN->info< CN->info)   CN = Cn->llink;
Else   CN = CN->rlink;
}
If(NN->info<PN->info)  Pn->llink = NN;
Else  PN->rlink = NN;
Return R;
}
```

```
NODE Search(Root, key)
```

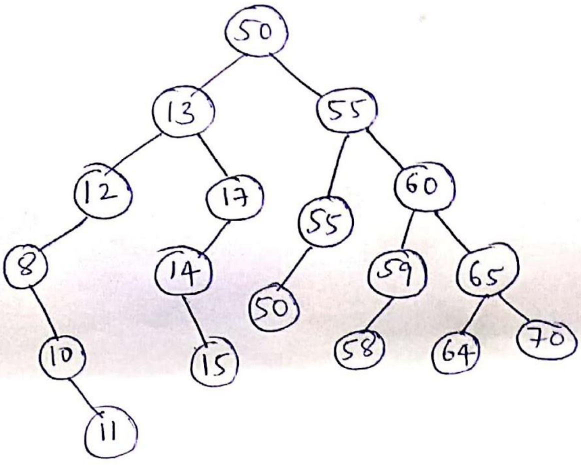
```
{
      Node NS=null;
    If(Root = NULL) return nULL;
If(R->info == key)
NS = R;
If(NS == NULL)
{
 If(Root->info>key)
 NS = search(Root->llnk, key);
Else
NS = search(Root->rlink , key)
}
Return NS
}

NODE Maximum(Root)
{
   NODE RN = Root;
   While(RN->rlink!=NULL)
    RN=RN->rlink;
  Return RN;
}
```

```
Void count(NODE Root,  int *cnt )


  If(Root==NULL) return;
  Coutn(Root->llink, cnt)
 *cnt++;
Count(Root->rlink, cnt);
}


Void countLeaf(NODE Root,  int *cnt )
{
   If(Root==NULL) return;
  CoutnLeaf(Root->llink, cnt)
If(Root->llink==NULL && Root->rlink==NULL)
 *cnt++;
CountLeaf(Root->rlink, cnt);
}
```

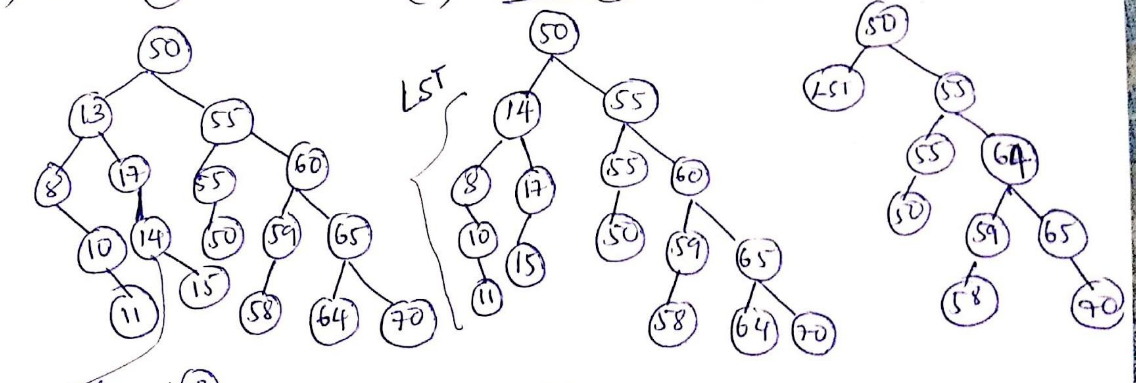
(1)

Construct the Binary Search tree for the sequence of i/p 50, 13, 17, 14, 55, 60, 12, 8, 65, 70, 55, 50, 10, 15, 11, 59, 58, 64

Write the updated trees for the following sequence of Deletions. (i) Delete 12 (ii) Delete 13 (iii) Delete 60 (iv) Delete Root (50)

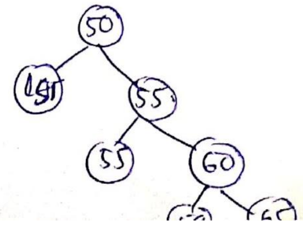


(i) Delete (12)   (ii) Delete (13)   (iii) Delete (60)



Inorder successor of (13)

(iv) Delete Root (50)


50
LST
55
55
60