**CS 3050 Homework # 4. Name :**

**Submitted to Blackboard, due at 11:59pm on March 16, 2017.**

1. Write pseudocode for RIGHT-ROTATE by modifying the algorithm in Figure 13.3 of the textbook.

LEFT-ROTATE(T,x)

1 y<-right[x]

2 right[x]<-left[y]

3 p[left[y]]<- x

4 p[y]<-p[x]

5 if p[x]=nil[T]

6 then root[T]<-y

7 else if x = left[p[x]]

8 then left[p[x]]<-y

9 else right[p[x]]<-y

10 left[y]<-x

11 p[x]<-y

So RIGHT-ROTATE is

RIGHT-ROTATE(T,y )

1 x<-left[y] set x

2 left[y]<-right[x] turn x’s right subtree into y’s left subtree.

3 p[right[x]]<-y

4 p[x]<-p[y] link y’s parent to x

5 if p[y] = nil[T]

6 then root[T]<-x

7 else if y = right[p[y]]

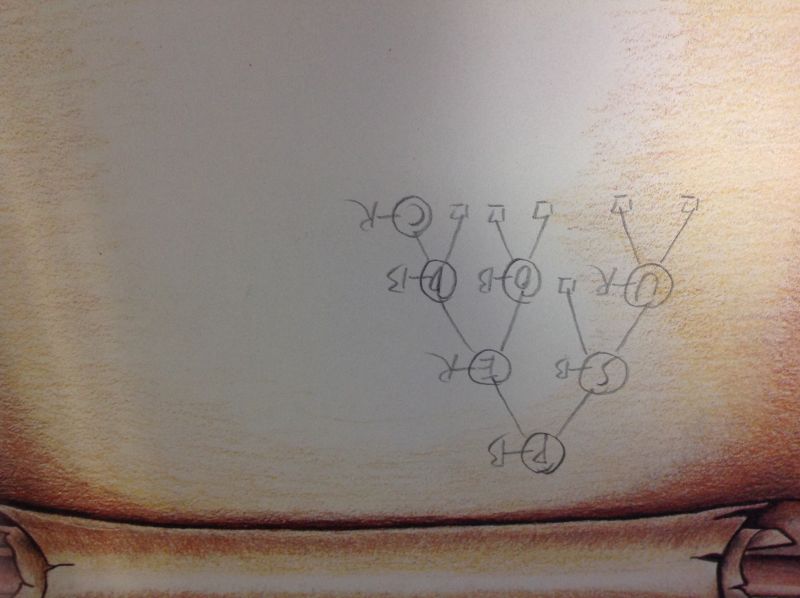
8 then right[p[y]]<-x

9 else left[p[y]]<-x

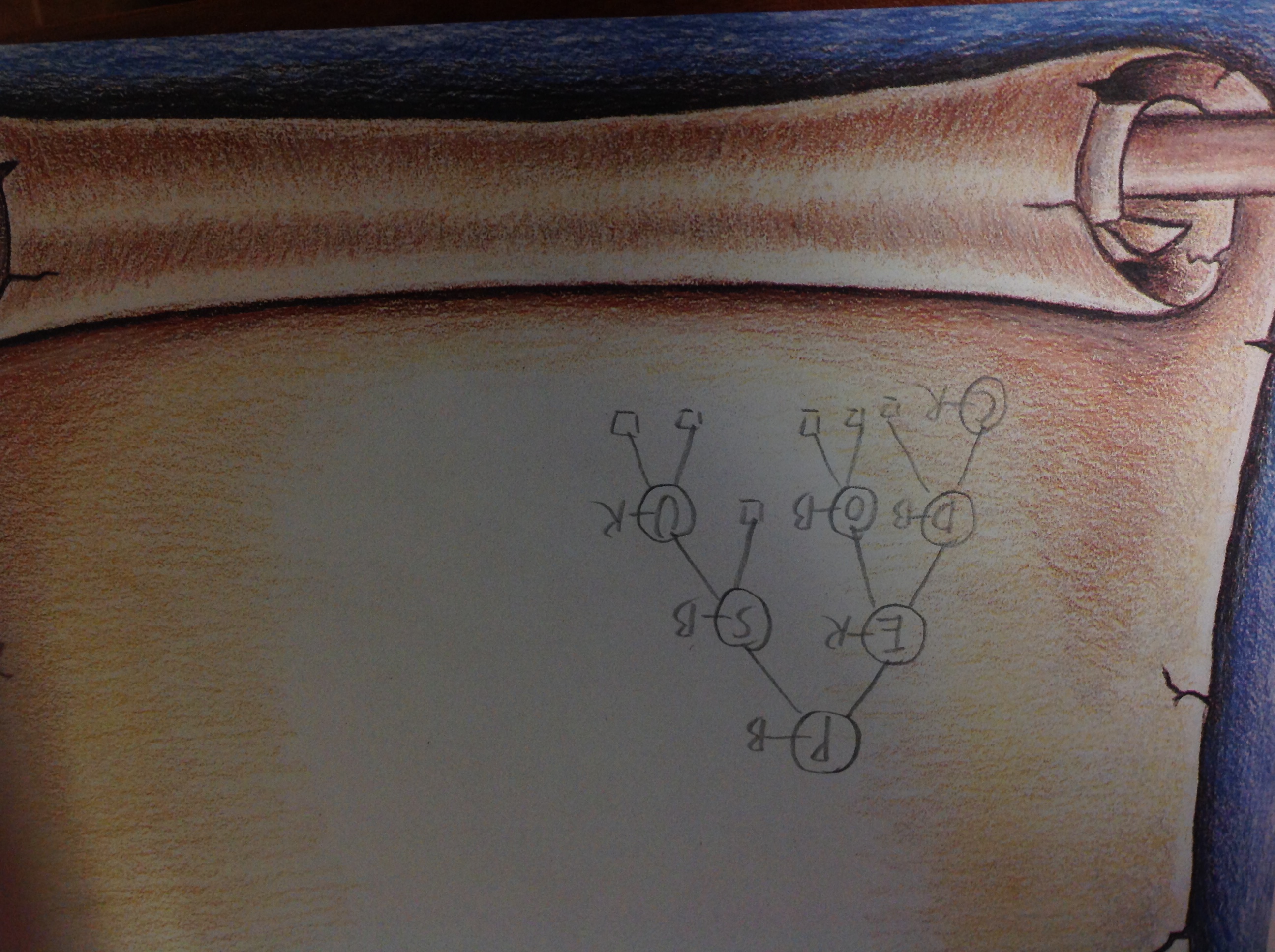
10 right[x]<-y put y on x’s right

11 p[y]<-x

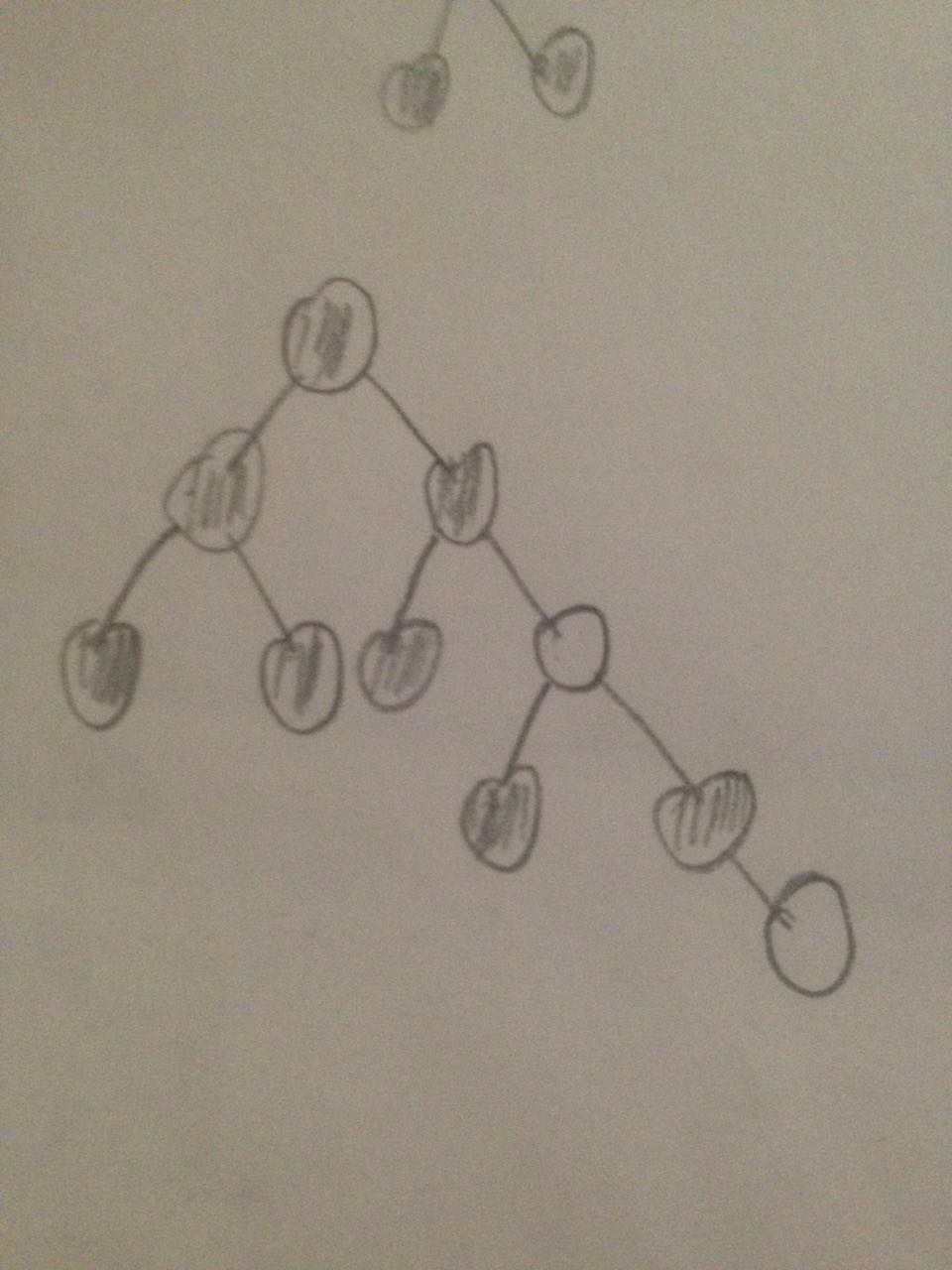
1. Draw the red-black trees that result after successively inserting the keys P, S, E, U, D, O, C into an initially empty red-black tree. The order of letters are alphabetical. You may use filled circles to indicate black and unfilled circles to indicate red.
2. if alphabetical is C>D>E>O>P>S>U



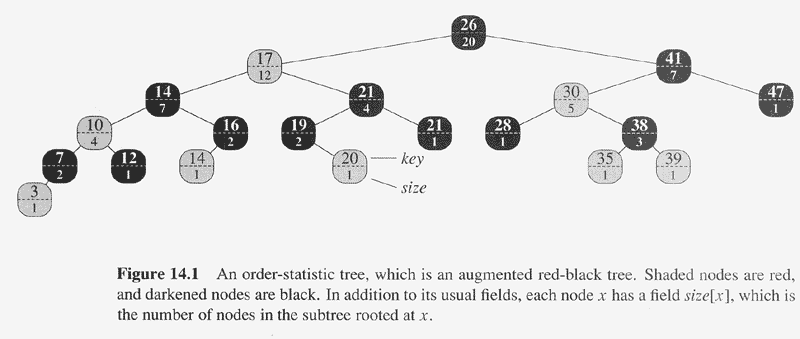
1. 2.if alphabetical is C<D<E<O<P<S<U；



3. Design a “worst-case” red-black tree with 10 nodes, i.e., a red-black tree with the longest possible path from the root to a leaf.



4. For the following order-statistics tree T, (1) show how OS-SELECT (T, 9) operates; (2) show how OS-RANK(T, 35) operates.



OS-SELECT(x，i)

1 r<-size[left[x]]+1

2 if i=r

3 then return x

4 elseif i<r

5 then return OS-SELECT(left[x],i)

6 else return OS-SELECT(right[x],i-r)

|  |  |  |
| --- | --- | --- |
| Time | key | i |
| 1 | 26 | 9 |
| 2 | 12 | 9 |
| 3 | 21 | 9-(7-1)=1 |
| 4 | 19 | I=r(return 19) |

OS-RANK(T,x)

1 r<-size[left[x]+1

2 y<-x

3 while y != root[T]

4 do if y = right[p[y]]

5 then r<-r+size[left[p[y]]]+1

6 y<-p[y]

7 return r

|  |  |  |
| --- | --- | --- |
| Times | Key[y] | r |
| 1 | 35 | 1 |
| 2 | 38 | 1 |
| 3 | 30 | 3 |
| 4 | 41 | 3 |
| 5 | 26 | 16 |

1. Given an interval tree T and an interval i , describe how to list all intervals in T that overlap i efficiently using pseudo code. [Hint: There are multiple ways to solve it. One way is to check both right and left subtrees from the root iteratively.]

Recursive way：

LIST\_OVERLAP(T, i)

if T=nil or i=0

then return

if left[T].high>T.low

then list “T.low,left[T].high”

If right[T].low<T.high

then list “right[T].low,T.high”

LIST\_OVERLAP(right[T], i/2)

LIST\_OVERLAP(left[T], i/2)

Iterative way：

LIST\_OVERLAP(T, i)

<queue> q

n <- T

while n is not empty

do {

if left[T].high>T.low

then list “T.low,left[T].high)”

if right[T].low<T.high

then print “right[T].low,T.high”

if n.left!=nil[T]

then q.push(n);

if right[n]!=nil[T]

then q.push(n);

n <- q.pop();

}