Sample solution
Exam 3
Spring, 2003

1:
pair : 3-6   key:2-5
Part a)

insert 110

```
                  (   50      90    )
                    /      |       \
              (20 30) (60 70) (100 110 120 130 160)
```

insert 140

```
          (   50      90      120     )
                    /      |       \         \
              (20 30) (60 70) (100 110) (130 140 160)
```

or

insert 140

```
          (   50      90       130     )
                    /      |       \           \
              (20 30) (60 70) (100 110 120) (140 160)
```

Part b)

Delete 30:
Step 1: Swap and delete

```
                  (   30      90    )
                    /      |       \
             (20)      (60 70)  (100 120 130 160)
```

Step 2: Restructure

```
                   (     90      )
                    /            \
             (20 30 60 70)  (100 120 130 160)
```

2.

```
             2
           /   \
         1       4
                   \
                    11
                  /    \
                6       12
              / \        \
             5    9       13
                /  \       \
               7    10     14
```

Part b)

after searching for a key 8., create a new node q, set q -> key
to some value other than i to indicate that there is no record in a tree
with key 8, and then insert q into the search external node.
(i.e., id the search stoped where p->LeftChild = 0,
insert q into p->LeftChild).
Because of splays, q will become the new root of a tree

```
      S                      B
      \                     /
        4                 11
      /                     \
     2                       12
   /    \                      \
  1      3                      13
                                  \
                                   14

      S                        B
      \                       /
       4                    11
     /    \               /    \
    2      6             9      12
  /   \   /               \       \
 1     3 5                 10      13
                                     \
                                      14

     S                          B
      \                        /
```

```
          4                        11
        /   \                    /   \
       2      6                 9      12
     /  \   / \                  \       \
    1     3 5   7                 10       13
                                             \
                                              14
```

3.

(a)

```
                 1001                    0010
    a:0:0001     ===>     a:0:0001       ===>          a:0:0001
      /                      /                            /
     a                  b:1:1001                   b:1:1001
                         /    \                      /     \
                        a       b               c:3:0010     b
                                                 /      \
                                                a         c
```

```
                    a:0:0001                             a:0:0001
    1000              /                  1010              /
    ===>         b:1:1001                ===>         b:1:1001
                   /     \                              /       \
              c:3:0010   d:4:1000                 c:3:0010        e:3:1010
               /      \   /      \                 /      \        /      \
              a         c d        b               a         c   d:4:1000    e
                                                                  /      \
                                                                 d         b
```

```
                         a:0:0001
    0100                   /
    ===>             b:1:1001
                      /        \
                f:2:0100        e:3:0010
                 /        \    /          \
    c:3:0010          f  d:4:1000    e
    /           \          /           \
   a              c       d              b
```

(b) Let p and q be the node to be deleted and the node has a back pointer to p, respectively. Then "b" is p and "d" is q.
Replace key of b by the key of d and change pointers appropriately.

```
                 a:0:0001
                  /
            b:1:1000
           /        \
      f:2:0100      e:3:1010
      /       \    /        \
 c:3:0010      f  b          e
 /        \
a          c
```

4.

(a) Suffix Trees

First, combine the two strings into one as follows:

ST=ST, where special characters $ and # are appended to each string.

Second, construct suffix tree of concatenated string ST.

This takes linear time, O(m+n)

While constructing suffix tree, we add information, length, substring to each internal node. The LCS is a substring with maximum length.

Example) S = "abc", T = "bcd"

ST=abc$bcd#

S1=abc$bcd#
S2=bc$bcd#
S3=c$bcd#
S4=$bcd#
S5=bcd#
S6=cd#
S7=d#
S8=#

LCS: "bc"

(b) Segment Trees

Let x be the x-coordinate value of the vertical line.

starting from the root,
visit node whose range covers x, and
report all horizontal line segments in it;
do this until a leaf node is reached.

O(logn + s)
  where s is the number of horizontal line segments reported.

(c) Priority Search Trees

First, consturct a priority search tree of line end points.
Then, enumerateRectangle(x,infinity,y) operation.

(d) Quad-trees

Question asks how many black pixels are there in the image. Thus we just need to traverse the quadtree Q, at the same time keep a level number. If we reach a white node, we stop and report zero. If a black node, we report the number of pixels indicated by the level number (here we assume the partitions are uniform, so the number of pixels at each level can be computed quickly). If we happen upon a grey node, we recursively traverse its children, sum up the numbers reported, and report the sum.