1)

Part a)

Delete 30:

Step 1: Swap and delete

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

Step 2: Restructure

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

Delete 100:

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

Delete 130:

```
            (    70    )
                    /         \
              (20 30 60)     (90 130)
```

Part b)          Maxminum Number keys

```
          level 1 :      M-1
      level 2 :     (M-1)*M
          level 3 :     (M-1)*M^2
           ...........
          level L :     (M-1)*M^L
```

So the number of Maxminum number of keys on Level L  is (M-1)*M^L

Total Maxminum number of key  is M^L -1

2)

a: Bottom-Up splay tree.

```
      D                       ===>      F
       \                              /   \
        N                            D     N
        /                            \     /
       E               F              E   J
        \            /  \                /
         J      ==> E    J              G
        /       /                \
       H      F        F                  H
      /        \        \
     G   ==>    G        G
    /            \        \
   F              H        H
```

b: Top-Down splay tree.

```
      S      T      B
      |      |      |
      D      E      N
     /      / \      \
    1      2  J       8
             / \
            H   7
           / \
          G   6
         / \
        F   5
       / \
      3   4

      S      T       B
      |      |       |
      D      H       N
     / \    / \     / \
    1   E  G   6   J   8
       /  / \       \
      2  F   5       7
        / \
       3   4

      S       T          B
      |       |          |
      D       F          N
     / \     / \        / \
    1   E   3   4      J   8
       /              / \
      2              G   7
                      \
                       H
                      / \
                     5   6
```

(3)

Part (a)

```
   (0):0:0010    0011   (0):0:0010     0100    (0):0:0010
     /          ===>        /        ===>        /
   (0)                **(1):4:0011**              (2):2:0100
                         /  \                  /      \
                       (0)  (1)         (1):4:0011    (2)
                                           /    \
                                          (0)   (1)



     1010  (0):0:0010      1011      (0):0:0010
     ===>      /           ===>          /
         (3):1:1010                 (3):1:1010
             /      \                /        \
       (2):2:0100   (3)         (2):2:0100     (4):4:1011
           /     \                /        \       /   \
     (1):4:0011  (2)         (1):4:1011   (2)   (3)   (4)
        /     \                /     \
      (0)     (1)            (0)    (1)



     1111  (0):0:0010
     ===>      /
            (3):1:1010
             /        \
       (2):2:0100      (5):2:1111
           /     \       /       \
     (1):4:0011  (2) (4):4:1011  (5)
        /     \        /     \
      (0)     (1)    (3)     (4)
```

5. quadtree

(a) If the pixel [i,j] is to the NW,NE, SW, or SE of current partitioning line, go to NW, NE, SW, or SE child. Repeat this until a leaf node is reached. This leaf node is the corresponding node to the pixel.

(b) You can initialize all the leaf nodes of the quadtree using the scheme in the question (a). This takes O(NlogN) time where N(n^2) is the number of pixels in the image.
Then, all internal nodes can be initialized by just traversing in post-order using black, white, and gray colors which takes O(N).

So, the total time complexity is O(NlogN).

(c)

1. If root is a leaf, return.

2. Rotate the children of the root.

   new (NW, NE, SE, SW) = previous (NE, SE, SW, NW)

3. Recursively perform the rotation in the subtree of the root.