1.

a)

Internal: 1500
External: 1501

Let n be total number of nodes in complete tree, i be number of internal nodes and e number of external. From structure of tree, we know n = i + e and e = i + 1. We may substitute e into first equation to obtain n = i + (i + 1) which can be rearranged to e = (n+1)/2. We may then subtract this from n to obtain i = (n-1)/2.

b)

h = floor($\log_2$(3) + k)

For complete binary tree, we know n$\leq$ $2^{h+1} - 1$, and solving for inequality yields h = floor($\log_2$(n)). In this case, n = $3(2^k)$, and we may say h = floor($\log_2(3(2^k))$). Simplifying using properties of logarithm gives h = floor(k + $\log_2$(3))

c)

Case 1: long chain of single child nodes, where there is only one external node (last node) and rest are internal nodes
Min external = 1; max internal = n -1

Case 2: only internal node is the root and all children are external nodes
Max external= n - 1; min internal = 1

2.

```
          X
        /   \
      A       L
     / \     / \
    K   J   B   R
     \     /
      C   S
```

Preorder: X A K C J L B S R

Identified root by taking last element of postorder traversal. Then drew tree based on both outputs – if didn't know where to place a node based off of one output, switch to other and use both outputs (and their respective traversal orders) to see position of node.

3.

```
              5
            /   \
          7       6
        /   \   /   \
      12     9 13    11
```

Postorder: 1,2,9,7,13,11,6,5 (Not non-increasing)

4.

a)

```
45      12      67      41      30      32      58      38        step1


    29              52              5               34
45      12      67      41      30      32      58      38        step 2

    12              41              5               34             step 3
45      29      67      52      30      32      58      38        downheaping


            43                              3
    12              41              5               34             step 4
45      29      67      52      30      32      58      38


            12                              3
    29              41              5               34             step 5
45      43      67      52      30      32      58      38        downheaping


                            59
            12                              3
    29              41              5               34             step 6
45      43      67      52      30      32      58      38


                            3
            12                               5
    29              41               30             34             step 7
45      43      67      52        59     32       58      38        downheaping
```

removeMin() 4x

38
12 5
29 41 30 34 removed 3,
45 43 67 52 59 32 58 last node to root

5
12 30
29 41 32 34 downheaping
45 43 67 52 59 38 58

58
12 30
29 41 32 34 removed 5,
45 43 67 52 59 38 last node to root

12
29 30
43 41 32 34
45 58 67 52 59 38 downheaping

38
29 30
43 41 32 34 removed 12
45 58 67 52 59 last node to root

29
38 30
43 41 32 34 downheaping
45 58 67 52 59

59
38 30
43 41 32 34 removed 29
45 58 67 52 last node to root

30
38 32
43 41 59 34 downheaping
45 58 67 52 (final heap)

b)

```
            45

            45
         12                          need reorder

            12
        45      67

            12
        45      67
     41                              need reorder

            12
        41      67
     45

            12
        41      67
     45    30                        need reorder

            12
        30      67
     45    41

             12
        30       67
     45    41   32                   need reorder

             12
        30       32
     45    41   67

             12
        30       32
     45    41   67   58

             12
         30        32
     45    41    67    58
   38                                need reorder
```

```
           12
       30        32
    38    41    67    58
45

           12
       30        32
    38    41    67    58
45    29                                        need reorder

             12
        29           32
    30       41     67    58
45    38

               12
          29          32
    30        41     67    58
45    38    52

               12
          29          32
    30        41     67    58
45    38    52    5                             need reorder

               5
          12          32
    30        29       67    58
45    38    52    41

                 5
          12            32
    30        29         67    58
45    38    52    41    34                      need reorder

                 5
          12            32
    30        29         34    58
45    38    52    41    67
```

5
12 32
30 29 34 58
45 38 52 41 67 43

5
12 32
30 29 34 58
45 38 52 41 67 43 3 need reorder

3
12 5
30 29 34 32
45 38 52 41 67 43 58

3
12 5
30 29 34 32
45 38 52 41 67 43 58 59 final heap

5.

```
class StackwPQ<E>{
Private PriorityQueue <Pair<Integer, E>> p;
//ts assumed to be in min mode. If in max mode, just need change a few things

Private int key = 0;

Public StackwPQ(){
	This.p = new PriorityQueue();
}

Public void push( E val){
	p.add(new Pair(key, val);
	Key - -;//since stack is LIFO, last pushed is min
}

Public E pop(){
	If p.isEmpty()
		Throw exception or something
	Return (p.removeMin().getElement());
}

Public boolean isEmpty(){
	Return p.isEmpty();
}

Public E peek(){
	If (p.isEmpty()) throw exception
	Return p.min();
}

}
```