1. I. B
   II. C
   III. E
   IV. C
   V. C
   VI. E
2.
   I.

i	min	0	1	2	3	4	5	6	7	8	9	10
		22	15	36	44	10	3	9	13	29	25	11
0	5	22	15	36	44	10	3	9	13	29	25	11
1	6	3	15	36	44	10	22	9	13	29	25	11
2	4	3	9	36	44	10	22	15	13	29	25	11
3	10	3	9	10	44	36	22	15	13	29	25	11
4	7	3	9	10	11	36	22	15	13	29	25	44
5	6	3	9	10	11	13	22	15	36	29	25	44
6	6	3	9	10	11	13	15	22	36	29	25	44
7	9	3	9	10	11	13	15	22	36	29	25	44
8	8	3	9	10	11	13	15	22	25	29	36	44
9	9	3	9	10	11	13	15	22	25	29	36	44
10	10	3	9	10	11	13	15	22	25	29	36	44

II.The worst case input for insertion sort would be an array of n items in reverse order, the time complexity would be O(n^2).

Example: [5,4,3,2,1]

This inner loop of the sort would need to be executed for all the previous elements of the current index, because it swaps elements that are only one index apart. In this case, when the array is in reverse, this happens for every element.

III.

a is the array
Ex. a,0,0,1 are the arguments passed to the merge method.

	0	1	2	3	4	5	6	7	8
size=1	H	O	M	E	W	O	R	K	N
a,0,0,1	H	O	M	E	W	O	R	K	N
a,2,2,3	H	O	E	M	W	O	R	K	N
a,4,4,5	H	O	E	M	O	W	R	K	N
a,6,6,7	H	O	E	M	O	W	K	R	N
size=2									
a,0,1,3	E	H	M	O	O	W	K	R	N
a,4,5,8	E	H	M	O	K	N	O	R	W
size=4									
a,0,3,7	E	H	K	M	N	O	O	R	W
size=5									
a,0,7,8	E	H	K	M	N	O	O	R	W

IV.

```
                                        12
            2                                             15
  1                   4                           13                    20
                 3          5                           14       18
                                6
                                    9
```

3. mistery(Key key) returns the ceiling if found, or null if the key is not in the BST.

4. Q F Y I O R A N H Z P X

I.

```
                  A
          H               F
    Q          O      R        X
 Y         P
```

i	0	1	2	3	4	5	6	7	8	9	10	11
a[i]		A	H	F	I	O	X	R	Q	N	Z	P

II.

```
                    I
          N                 R
    Q          O        X        Z
 Y         P
```

i	0	1	2	3	4	5	6	7	8	9
a[i]		I	N	R	Q	O	X	Z	Y	P

5.

The data type will consit of three fields: a min heap, a maxheap and the current median. The min heap will contain keys that are bigger than the median, and the max heap will contain the keys smaller than the median.

```
getMedian():
    return the current median

insertKey( key):
    check if median exists, if not the new key will become the median
        else compare key with current median:
         if key < median:
              add key to max heap
         else if key >= median:
              add key to min heap
    if size of max heap > size of min heap:
         add median to min heap
         the new median will be the first element of the max heap.
         remove from max heap
    else if min heap > max heap + 1:
         add median to max heap
         the new median will be the first element of the max heap.
         remove from max heap

removeKey():
    if size of max heap >= size of min heap:
         make new median the root of the max heap
    else
         make new median the root of the max heap

    return old median
```

Complexities:

getMedian: this one is clearly O(1) as we keep it as a field of the ADT.

insertMedian:On the worst case the key is added to one of the heaps, which has a time complexity of O(logn), and then if the heap sizes are not balanced the median is added to one of the heaps, O(logn) and a the root is removed from the other heap

removeMedian:in the worst case, the new median is added to one of the heaps which has a worst time complexity of O(logn)