Consider a set of elements {12,34,56,73,24,11,34,56,78,91,34,91,45}. Sketch the **heapsort** algorithm and use it to sort this set. Obtain a derivation for the time complexity of heapsort, both the worst case and average case behaviour.

Heapsort (worst case complexity T(n) = O(n log n)) uses a data structure known as a max-heap which is a complete binary tree where the element at any node is the maximum of itself and all its children. A tree can be stored either with pointers as per the pictorial representation we are normally used to visualize, or by mapping it to a vector. Here if a node in the tree is mapped to the ith element of an array, then its left child is at 2i, its right child is at (2i+1) and its parent is at floor(i/2).

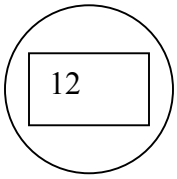
We can construct a heap by starting with a an existing heap, adding an element at the bottom of the heap, and allow it to migrate up the father chain till it finds its proper position. **Complexity *T(n) = O(n log n)***.

LET US BUILD THE HEAP BY SIMULATION USING INSERT

The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

1. Start with the element 12 and obtain a one element heap.

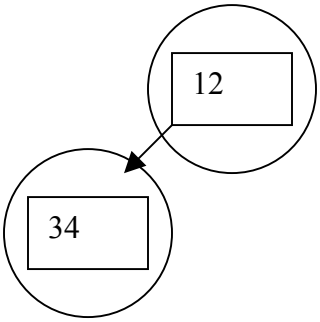
1												
12												



2. Consider the next element 34 and add it to the bottom of the heap.

1. Start with the element 12 and obtain a one element heap.

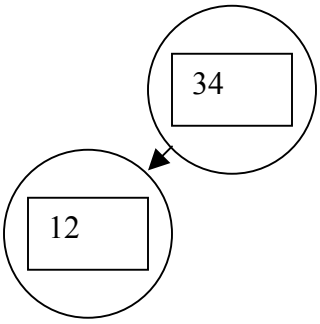
1	2											
12	34											



1a. The heap has to be adjusted and 34 inserted at the right place. We go up the father chain, 12 comes down and 34 goes up.

1. Start with the element 12 and obtain a one element heap.

1	2											
34	12											

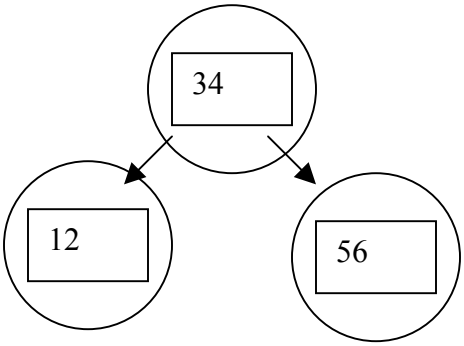


The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

3. The next element is 56 which is to be inserted into the heap. Add it to the bottom of the heap.

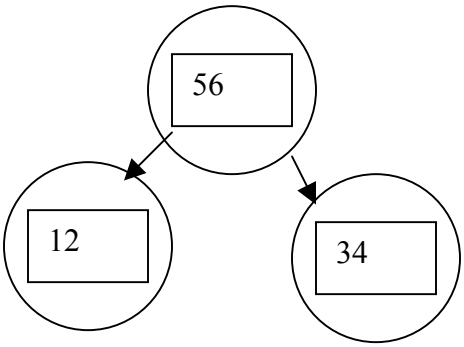
1. Start with the element 12 and obtain a one element heap. 56 has to move up the father chain to find its right place.

1	2	3										
34	12	56										



3b. Move 56 up and 34 down. This gives the resulting heap shown below.

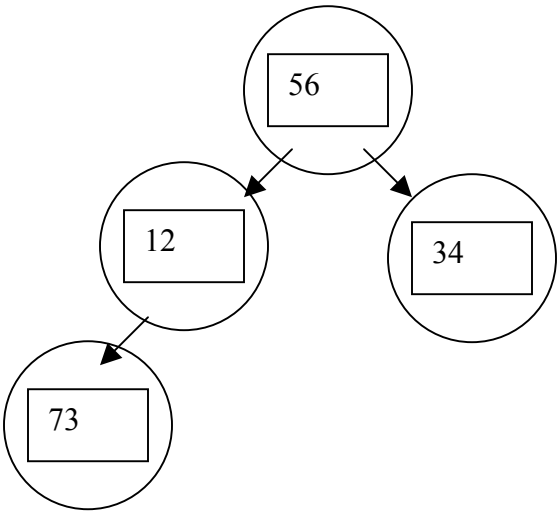
1	2	3										
56	12	34										



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

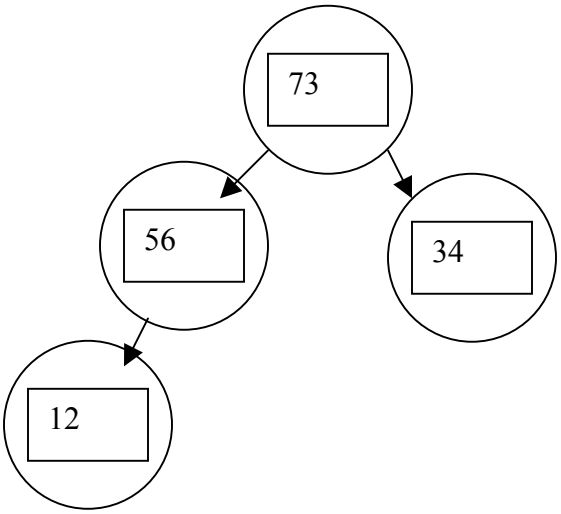
4. The next element is 73 which is to be inserted into the heap. Add it to the bottom of the heap.

1	2	3	4									
56	12	34	73									



4a. 73 is greater than its parent, so it has to move up, and 12 moves down. Then further 73 moves up and 56 moves down. This gives the heap shown below.

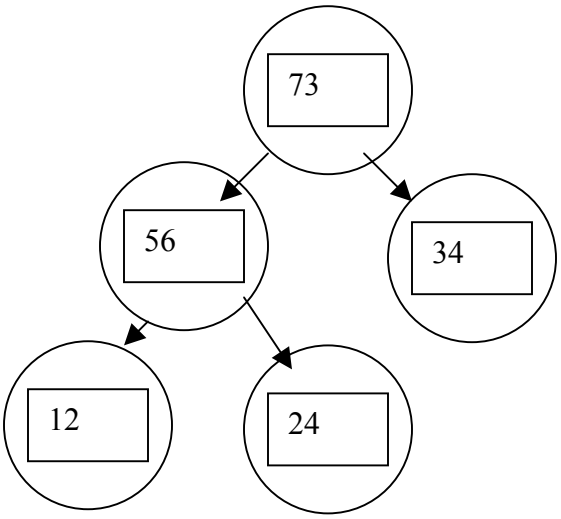
1	2	3	4									
73	56	34	12									



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

5. The next element is 24 which is to be inserted into the heap. Add it to the bottom of the heap.Since it is less than 56 we have a heap.

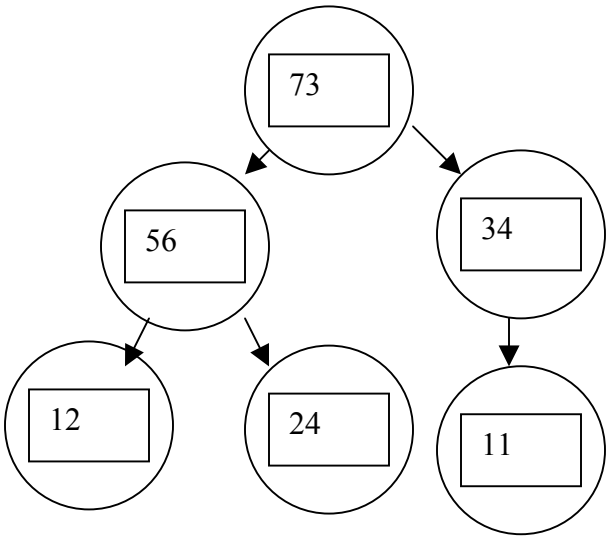
1	2	3	4	5								
73	56	34	12	24								



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

5. The next element is 11 which is to be inserted into the heap. Add it to the bottom of the heap.Since it is less than 34 we have a heap.

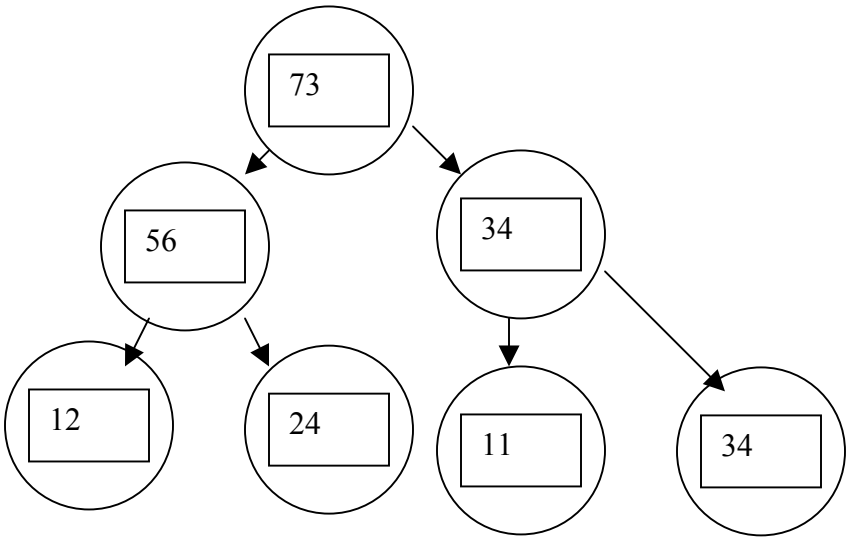
1	2	3	4	5	6							
73	56	34	12	24	11							



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

5. The next element is 34 which is to be inserted into the heap. Add it to the bottom of the heap.Since it is equal to its parent 34 we have a heap.

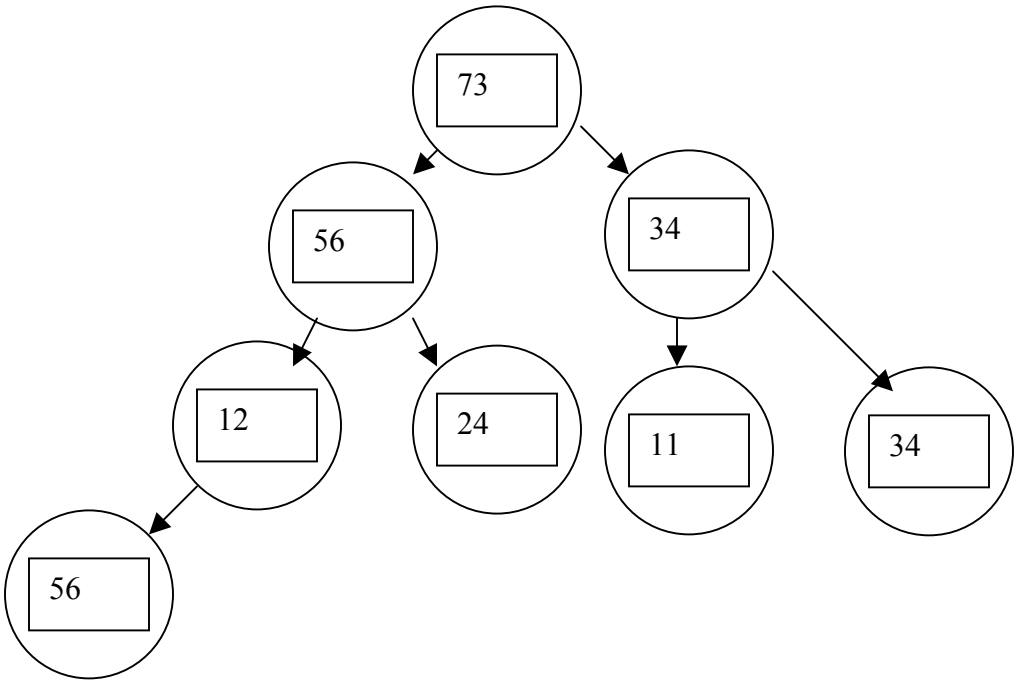
1	2	3	4	5	6	7						
73	56	34	12	24	11	34						



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

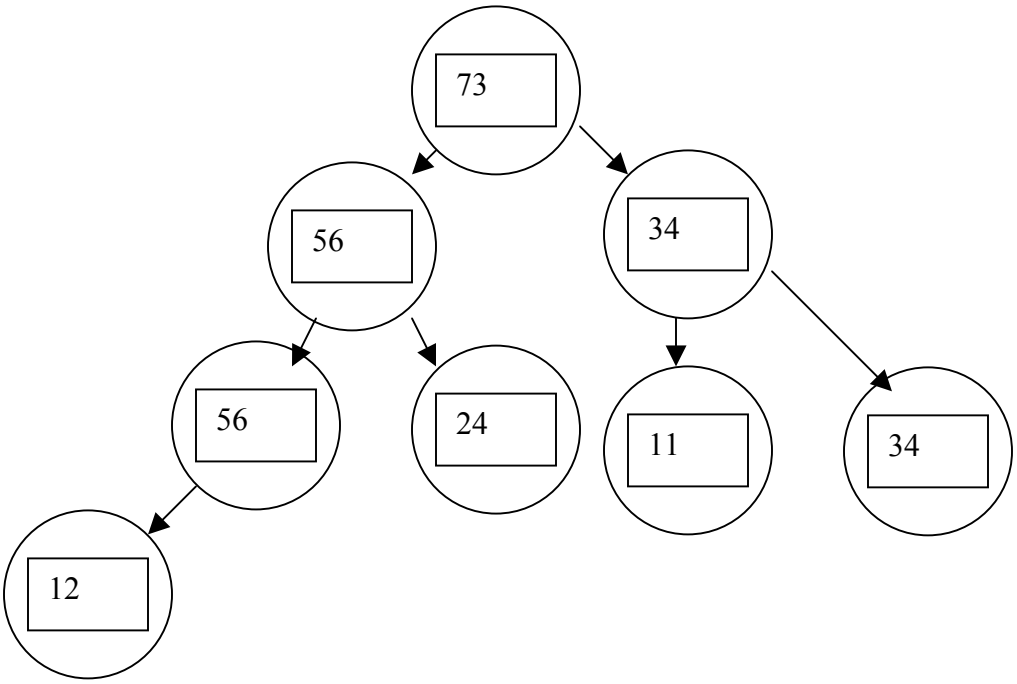
5. The next element is 56 which is to be inserted into the heap. Add it to the bottom of the heap.

1	2	3	4	5	6	7	8					
73	56	34	12	24	11	34	56					



5a. The heap has to be adjusted with 56 going up and 12 coming down.

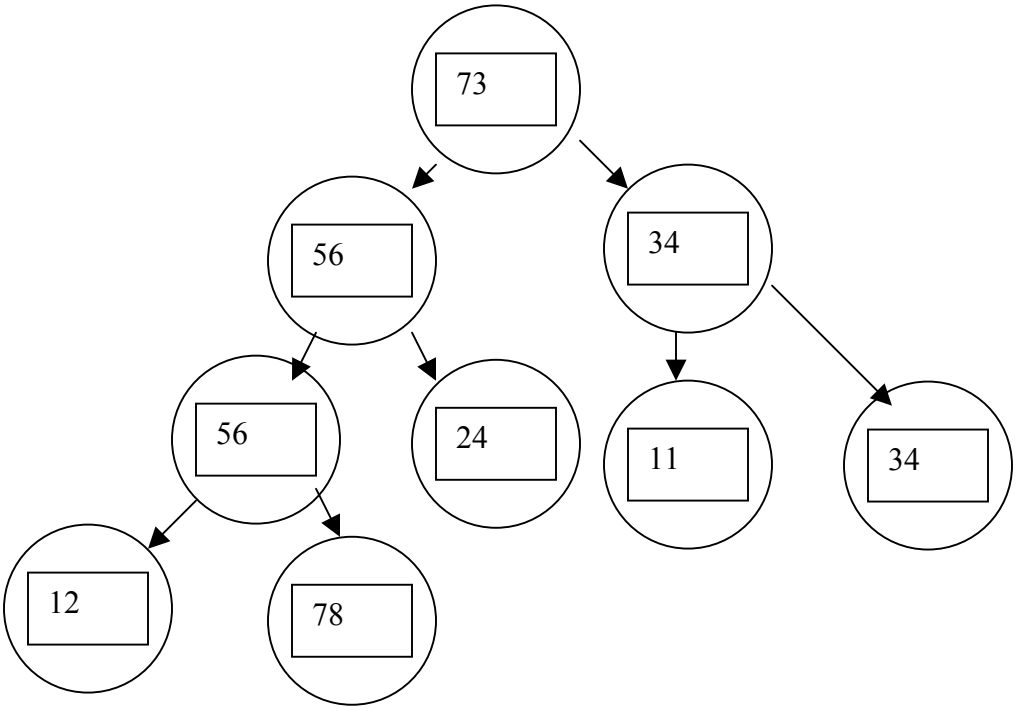
1	2	3	4	5	6	7	8					
73	56	34	56	24	11	34	12					



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

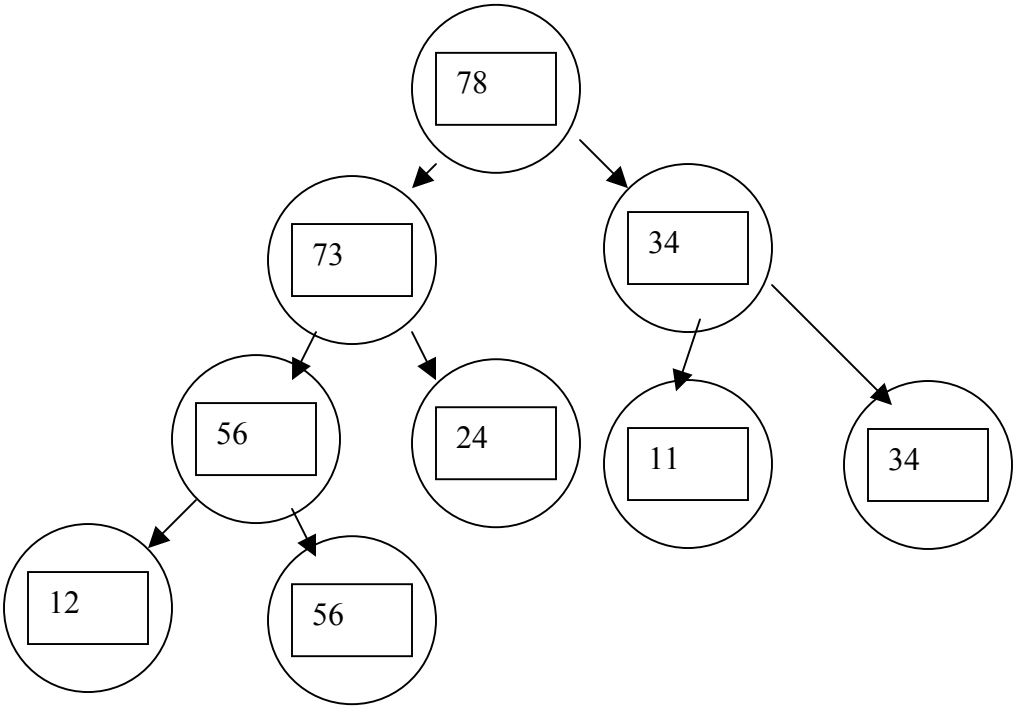
6. The next element is 78 which is added to the bottom of the heap. 78 has to be adjusted in the heap. Going up the father chain from 78 we see that all the elements in the father chain have to be moved down one position and 78 goes to the root.

1	2	3	4	5	6	7	8	9				
73	56	34	56	24	11	34	12	78				



6a. Properly positioning 78 we get.

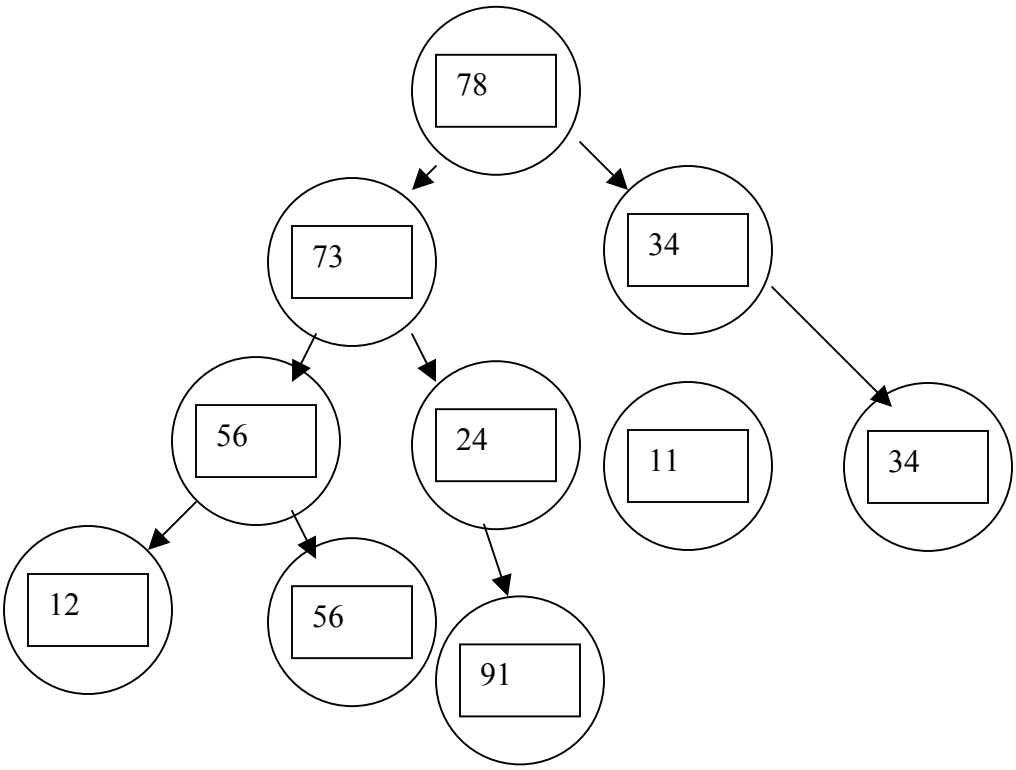
1	2	3	4	5	6	7	8	9				
78	73	34	56	24	11	34	12	56				



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}
7. The next element is 91 which is added to the bottom of the heap. 91 has to be adjusted in the heap. Going up the father chain from 91 we see that all the elements in the father chain have to be moved down one position and 91 goes to the root.

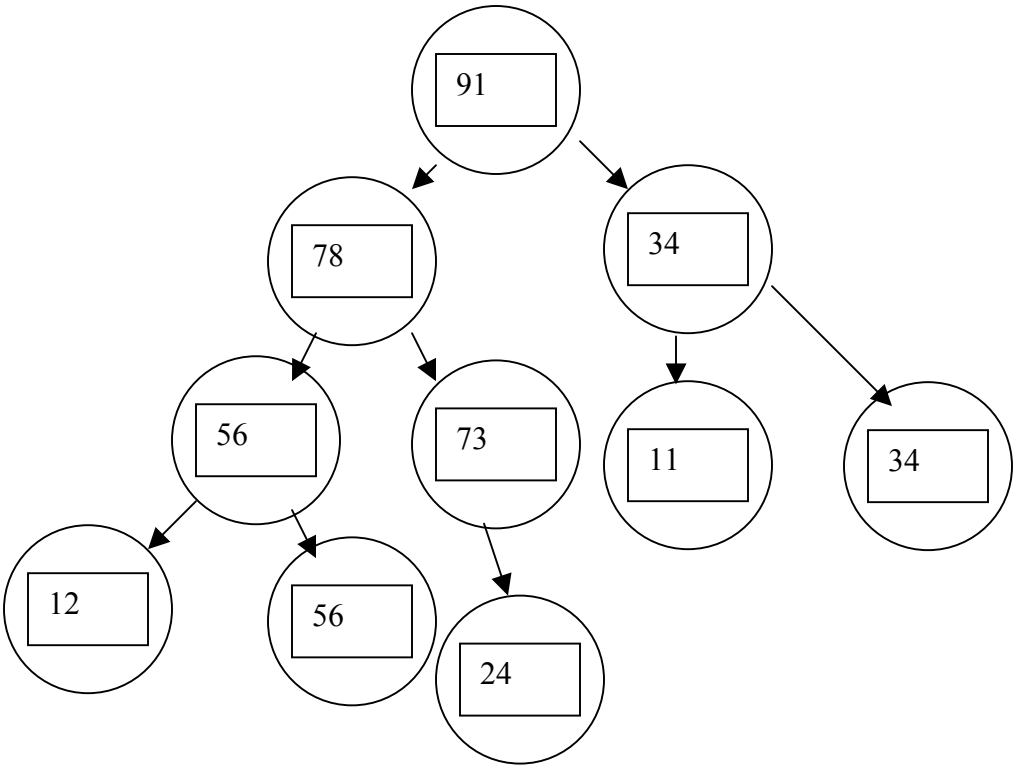
Inserting 91 we get.

1	2	3	4	5	6	7	8	9	10			
78	73	34	56	24	11	34	12	56	91			



7a. Positioning 91 correctly we get

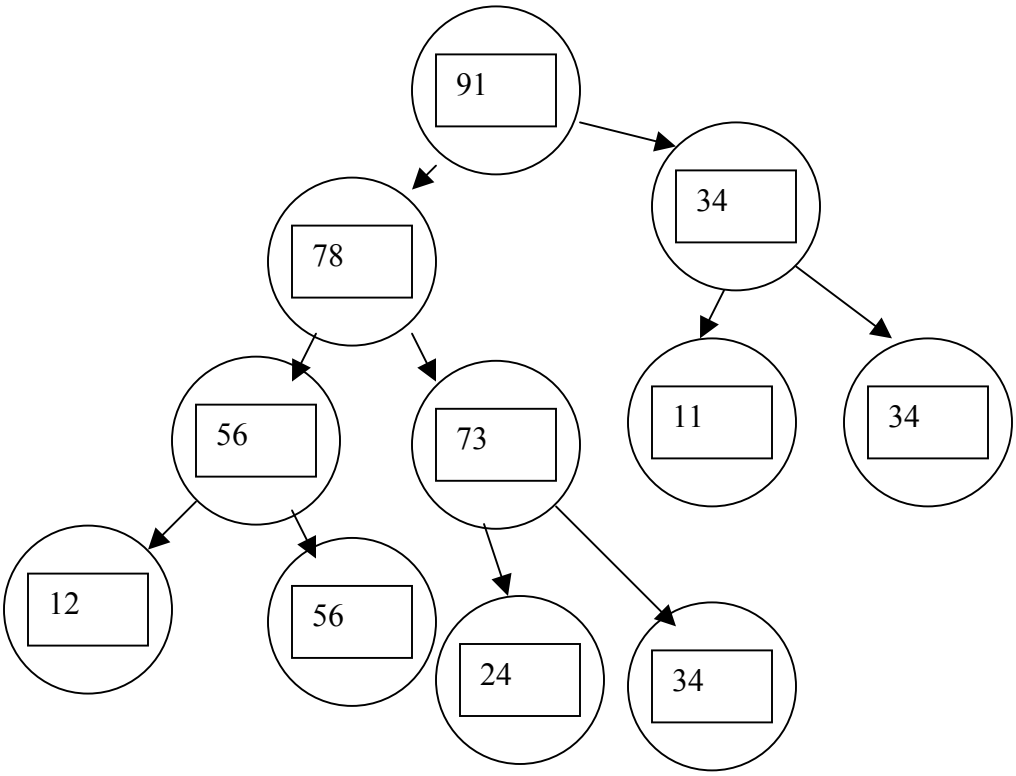
1	2	3	4	5	6	7	8	9	10			
91	78	34	56	73	11	34	12	56	24			



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

8. The next element is 34 which is added to the bottom of the heap. The heap property is maintained.

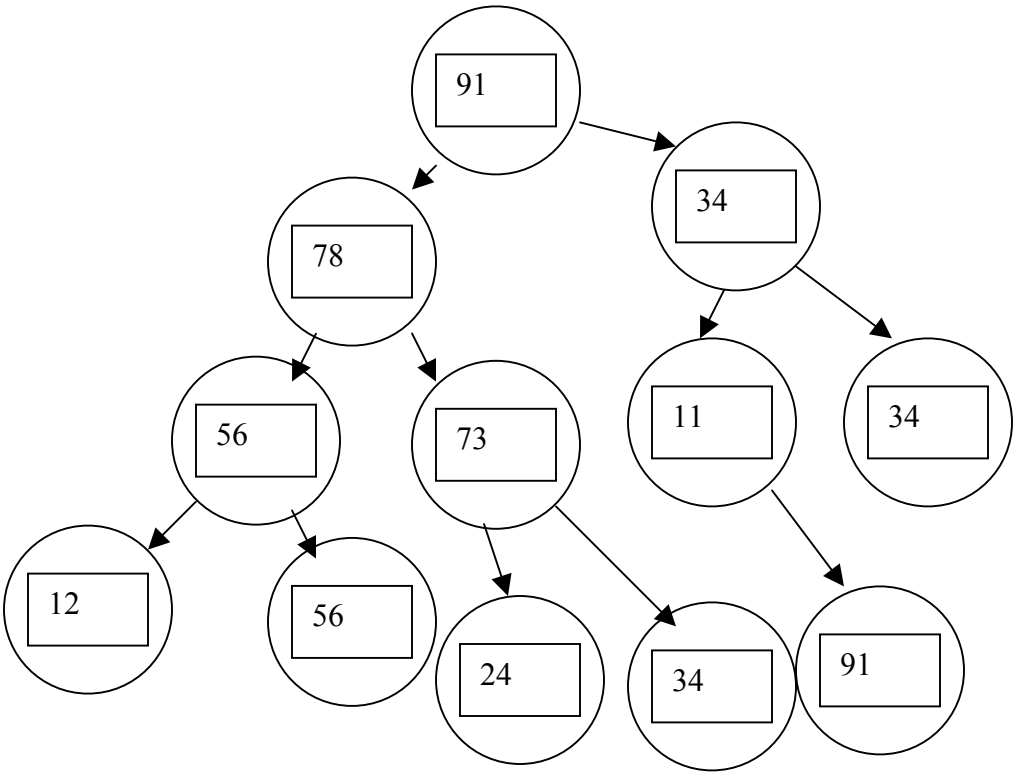
1	2	3	4	5	6	7	8	9	10	11		
91	78	34	56	73	11	34	12	56	24	34		



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

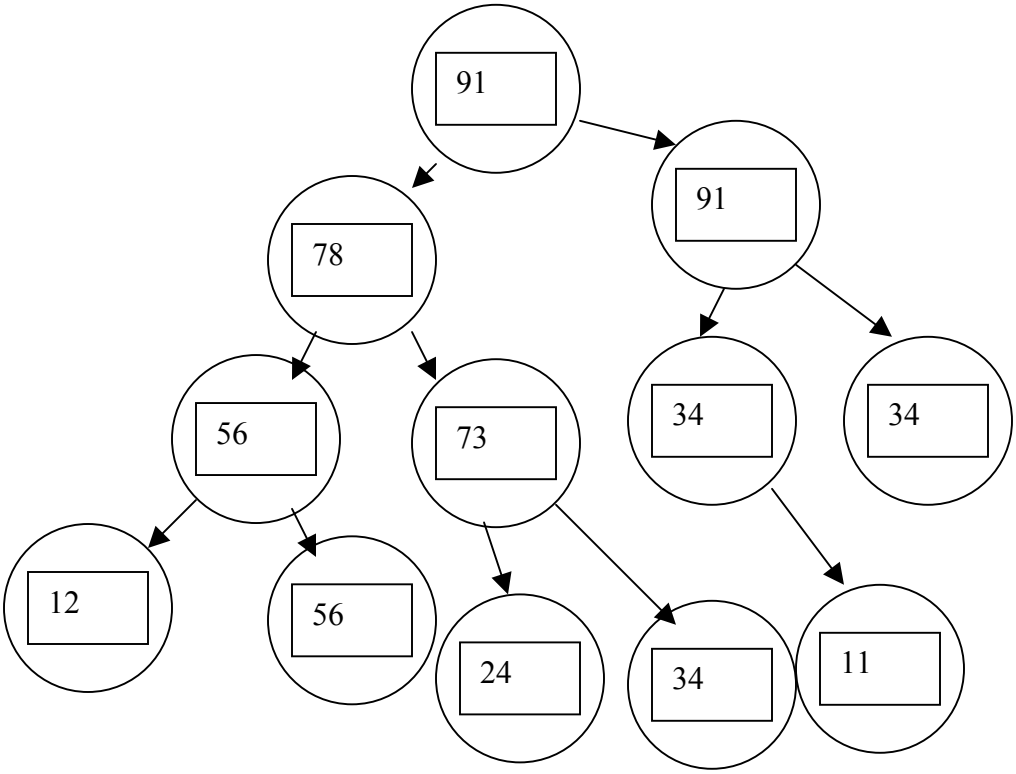
9. The next element is 91 which is added to the bottom of the heap. The heap property is not maintained.

1	2	3	4	5	6	7	8	9	10	11	12	
91	78	34	56	73	11	34	12	56	24	34	91	



9a. To maintain the heap property 91 should go next to the root and every element in the father chain moves down one position.

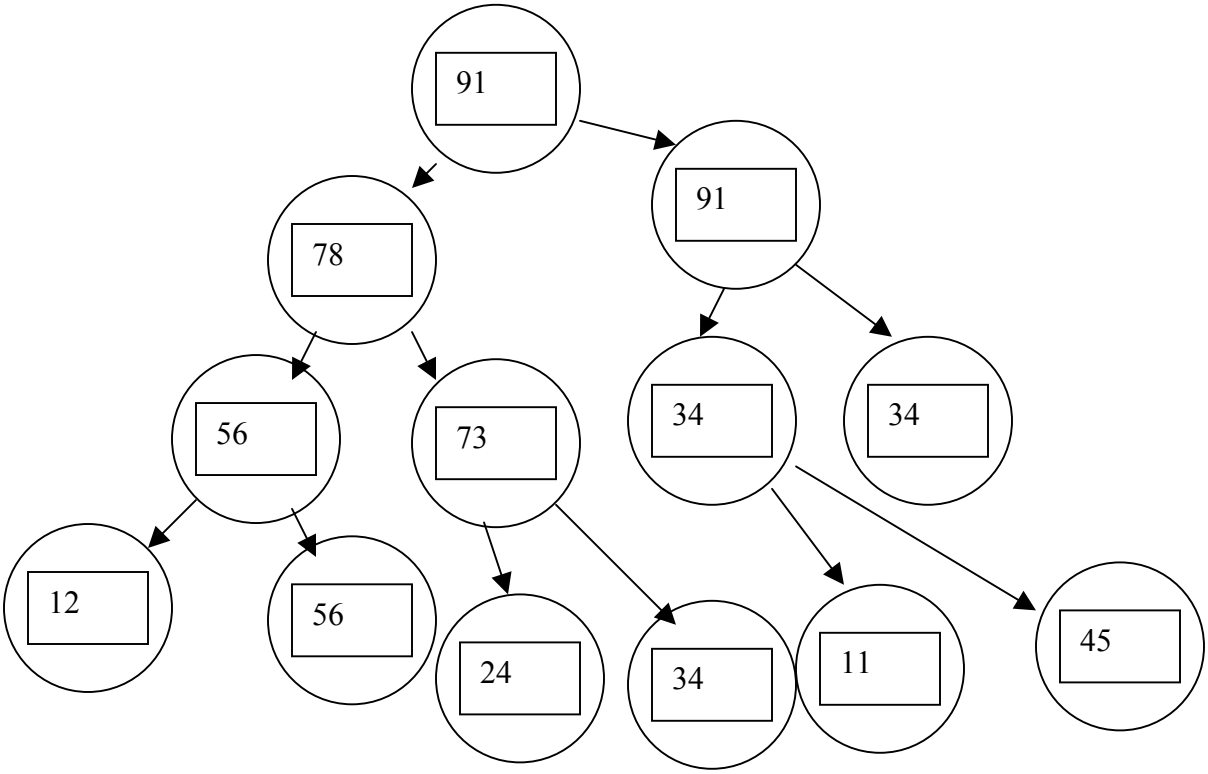
1	2	3	4	5	6	7	8	9	10	11	12	
91	78	91	56	73	34	34	12	56	24	34	11	



The set is {12,34,56,73,24,11,34,56,78,91,34,91,45}

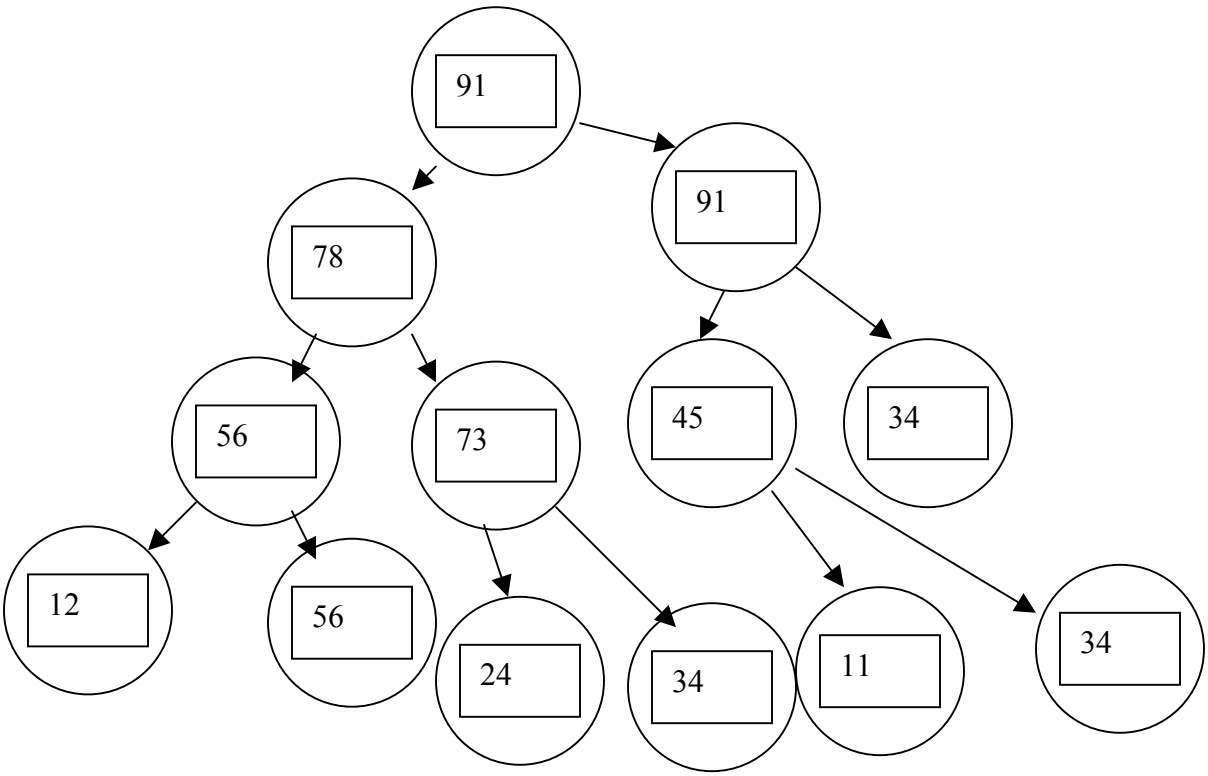
10. The next element is 45 which is added to the bottom of the heap. The heap property is not maintained. 45 should go up and 34 come down to obtain the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
91	78	91	56	73	34	34	12	56	24	34	11	45



10a. Adjusting the heap we get the **final heap**.

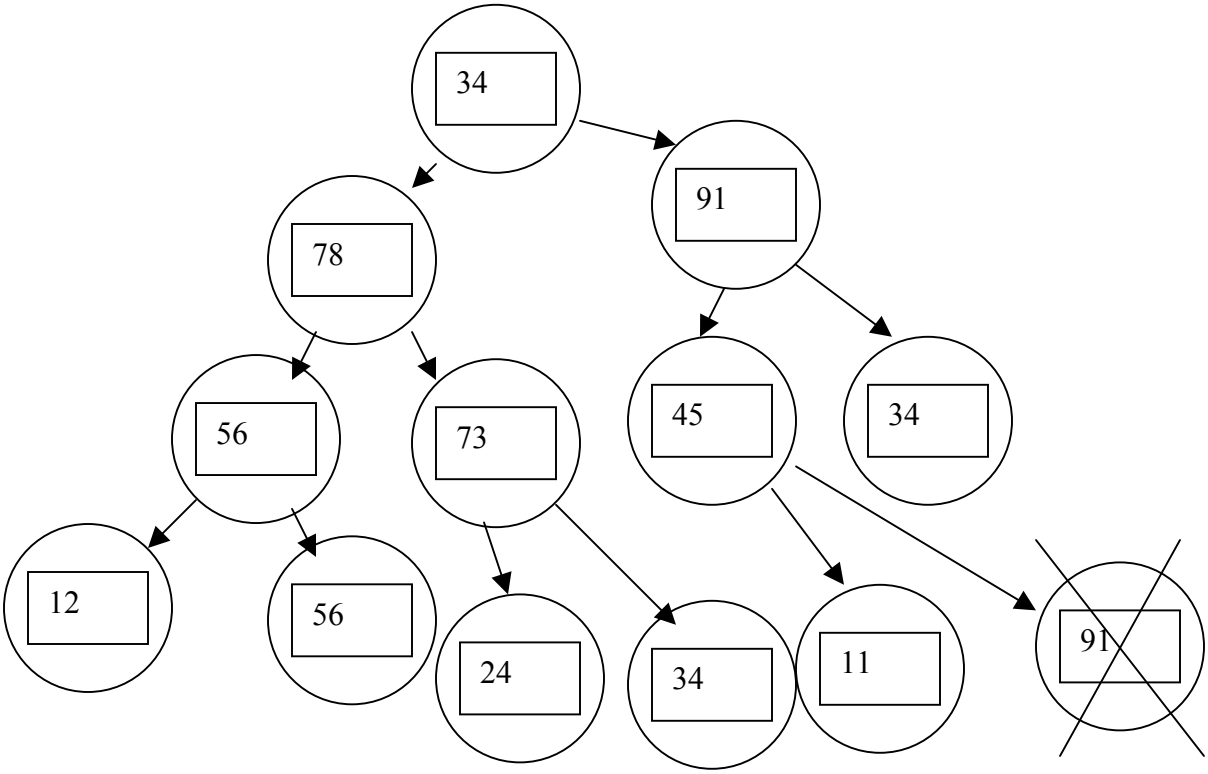
1	2	3	4	5	6	7	8	9	10	11	12	13
91	78	91	56	73	45	34	12	56	24	34	11	34



NOW FOR THE HEAPSORT ALGORITHM.

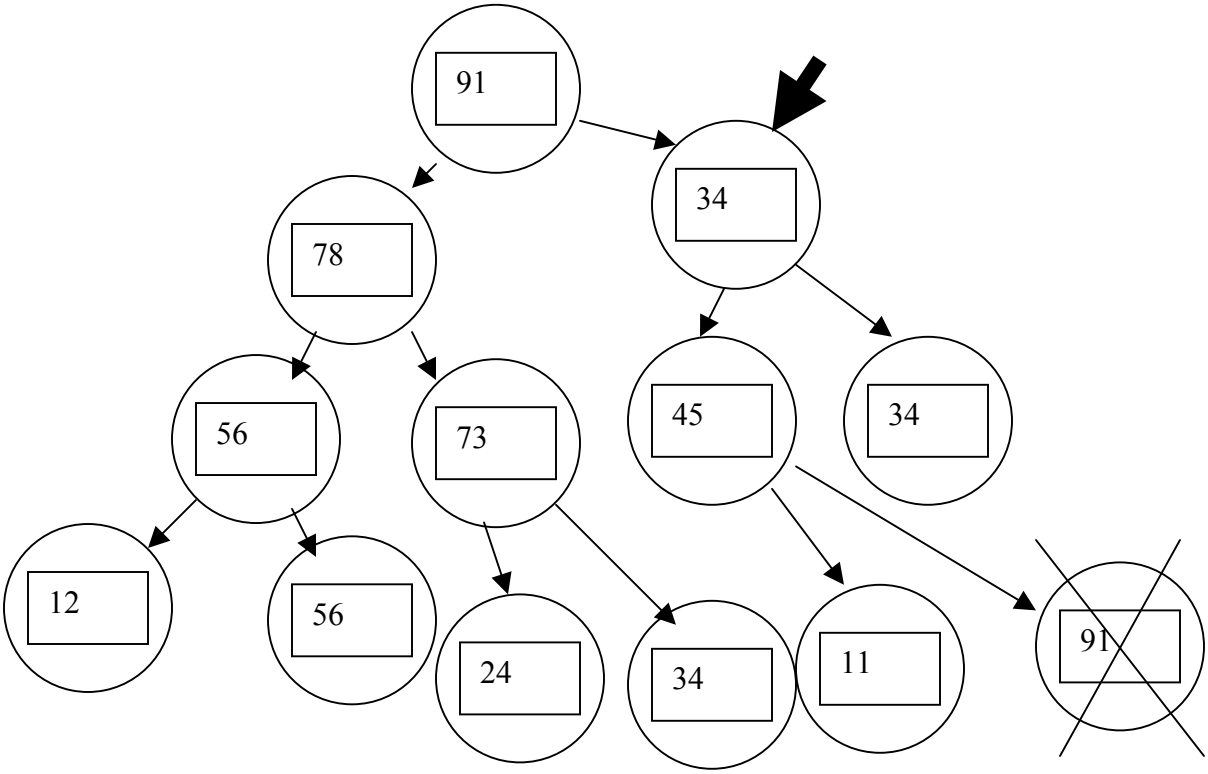
1. Delete the maximum element, i.e. the root of the heap by swapping it with the element at the bottom of the heap, and ignore the element at the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
34	78	91	56	73	45	34	12	56	24	34	11	91



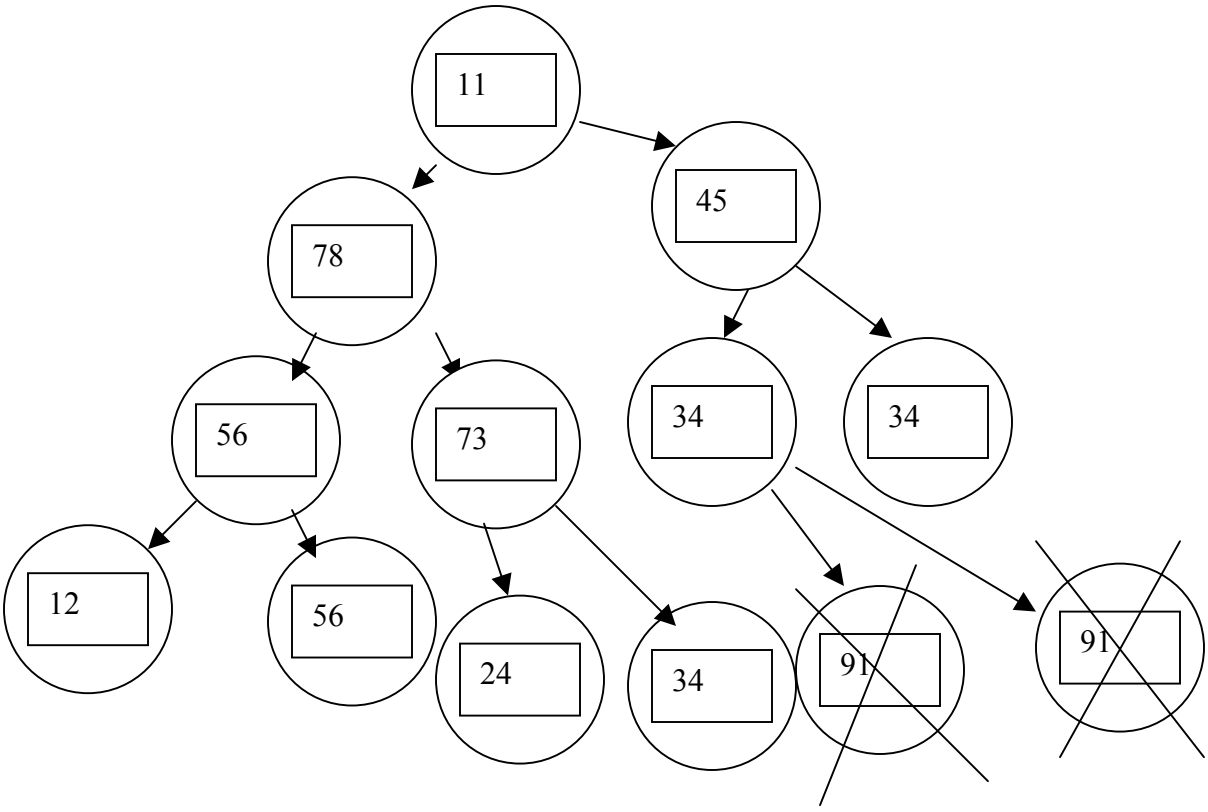
Now the heap property may be violated. So 34 at the root is compared with its two children, and 91 goes up and 34 comes down.

1	2	3	4	5	6	7	8	9	10	11	12	13
91	78	34	56	73	45	34	12	56	24	34	11	91



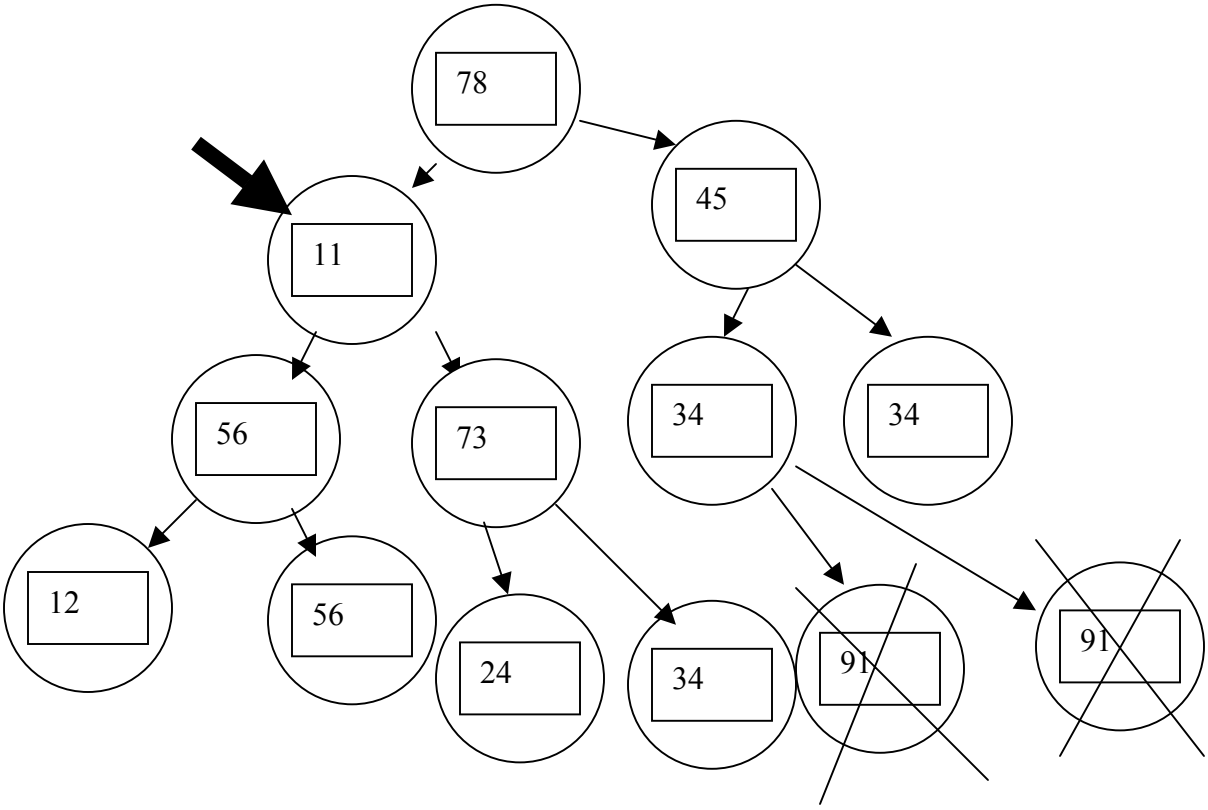
Now 34 is compared with its two children, 45 is larger so 34 moves down and 45 moves up. 34 now is greater than its child 11, so we have a heap.

11	2	3	4	5	6	7	8	9	10	11	12	13
11	78	45	56	73	34	34	12	56	24	34	91	91



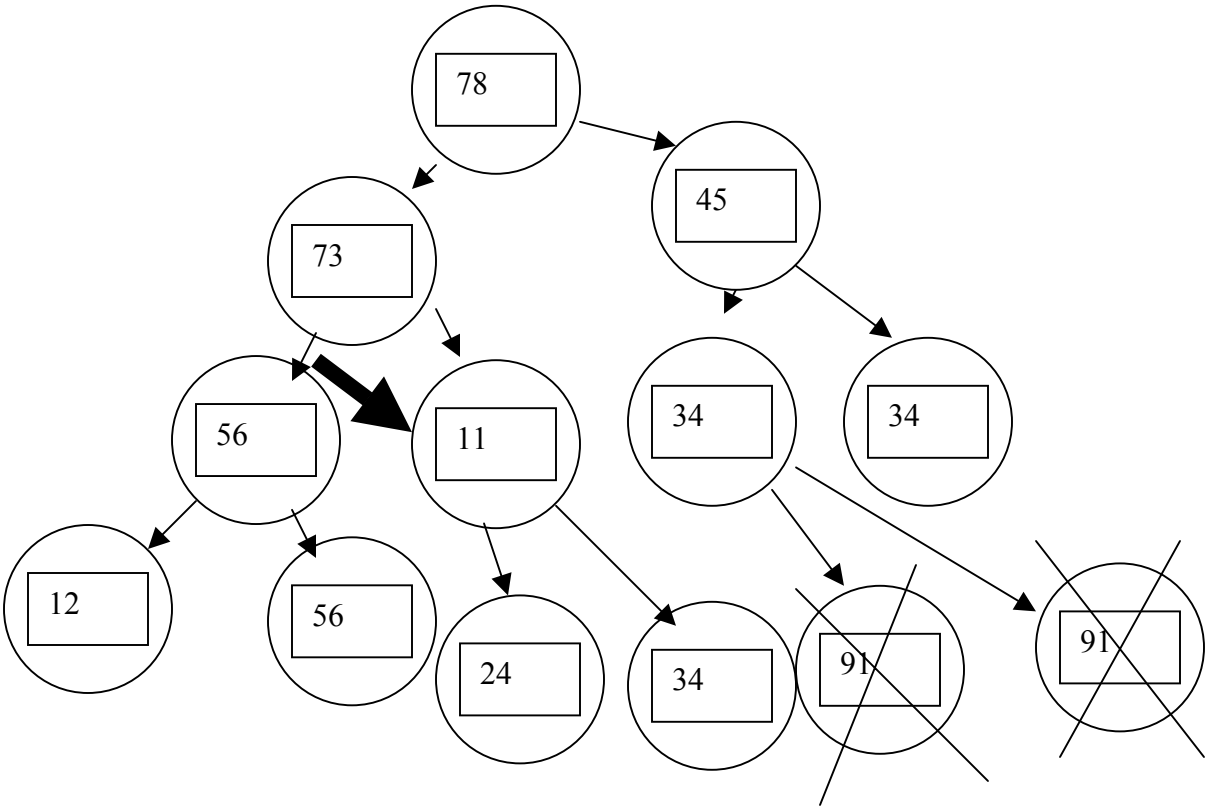
Now delete the maximum element, i.e the root of the heap, by swapping it with the element at the bottom of the heap and ignore the bottom of the heap The element 11 at the root has to be adjusted. Comparing it with its two children we find 78, the left child is larger. So 78 moves up and 11 moves down.

1	2	3	4	5	6	7	8	9	10	11	12	13
78	11	45	56	73	34	34	12	56	24	34	91	91



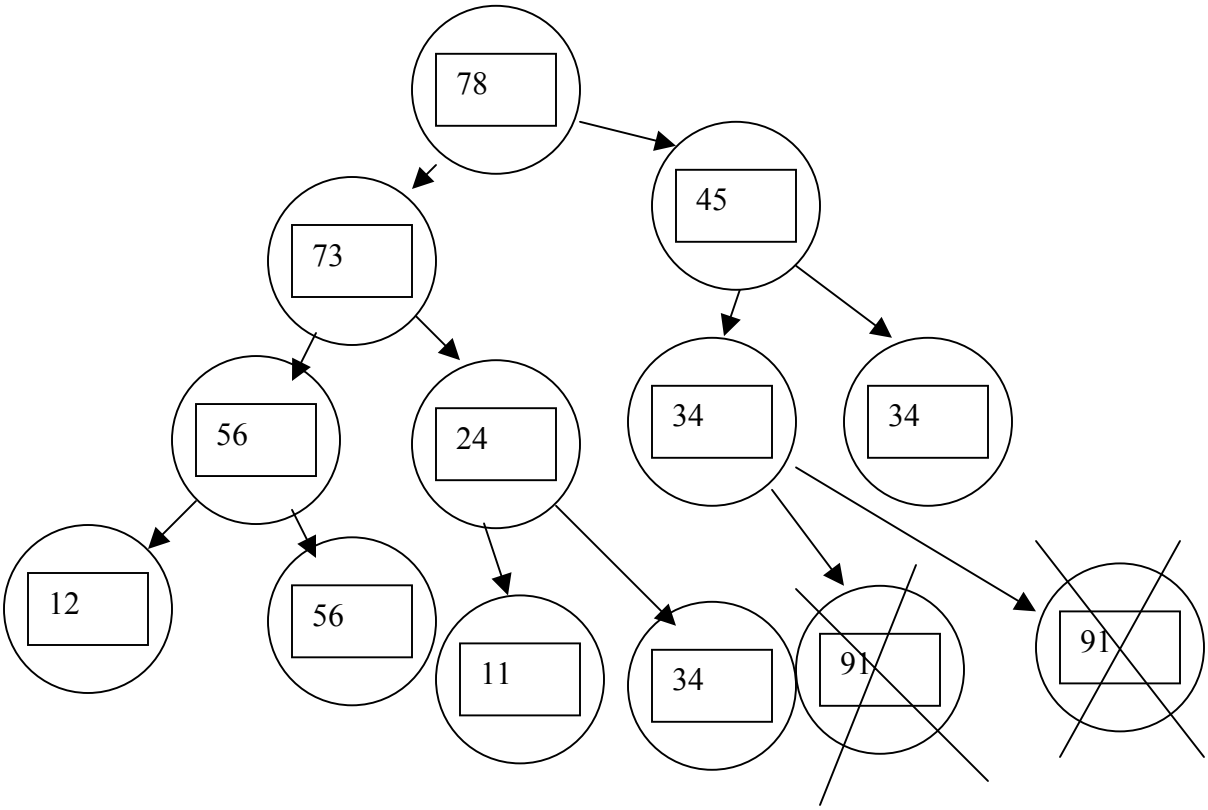
Now 11 is compared with its two children and we find 73 is larger, so 73 goes up and 11 comes down.

1	2	3	4	5	6	7	8	9	10	11	12	13
78	73	45	56	11	34	34	12	56	24	34	91	91



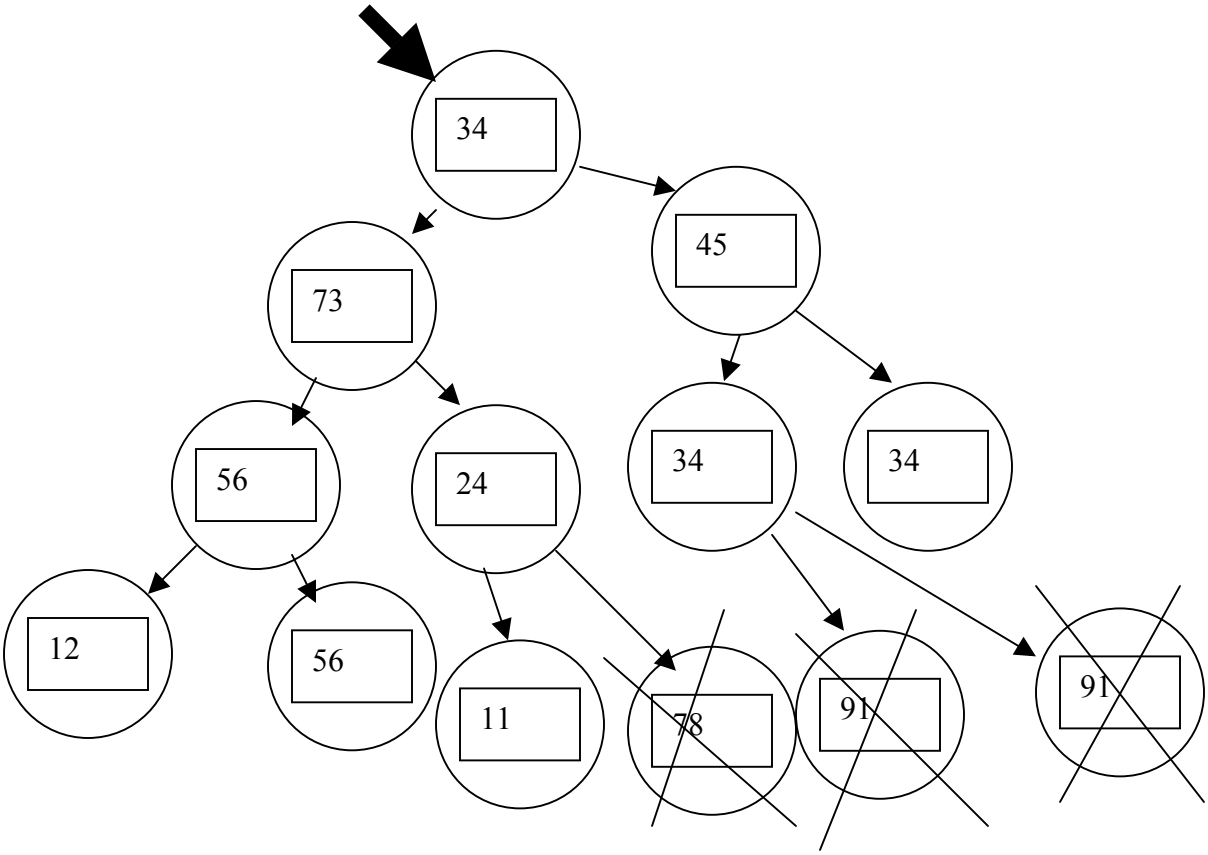
Now 11 is less than its child 24, so 24 goes up and 11 comes down. We finally have adjusted the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
78	73	45	56	24	34	34	12	56	11	34	91	91



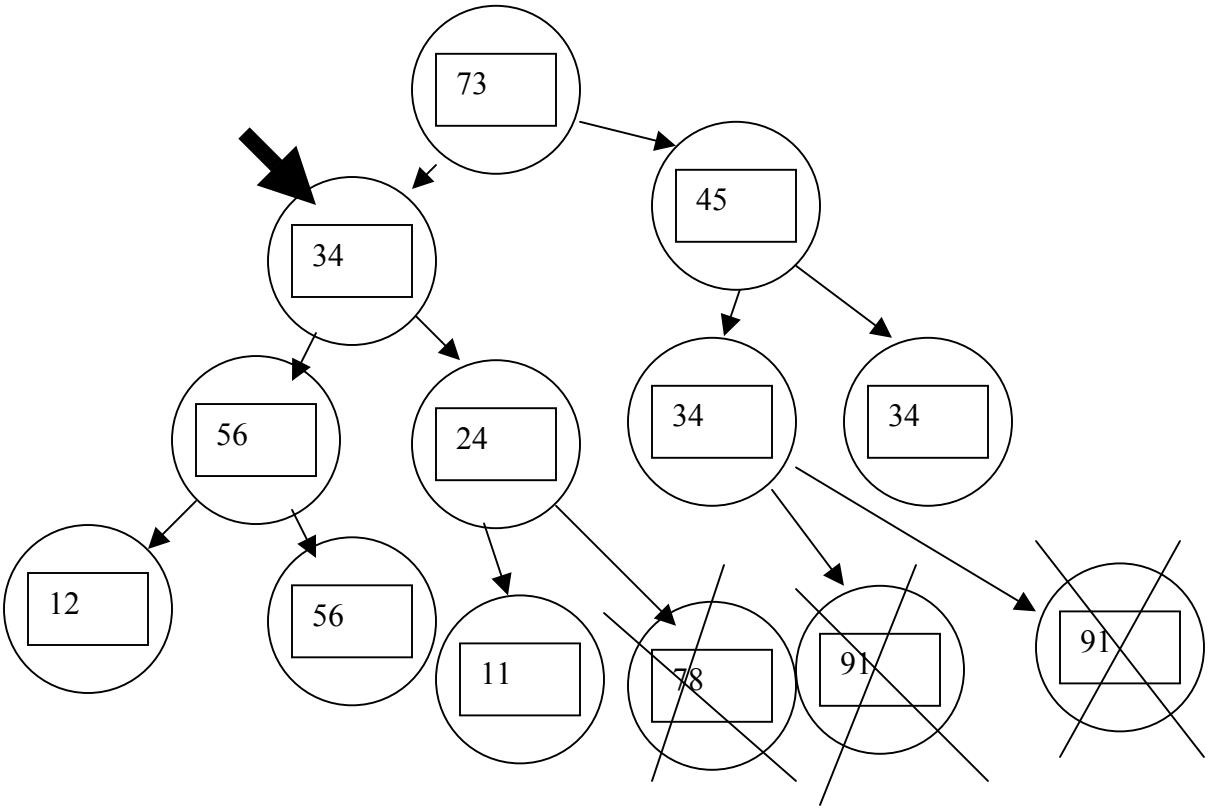
Now we delete the maximum element from the heap, by swapping the bottom of the heap with the root. So 34 comes to the top of the heap and 78 goes to the bottom, and then we ignore the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
34	73	45	56	24	34	34	12	56	11	78	91	91



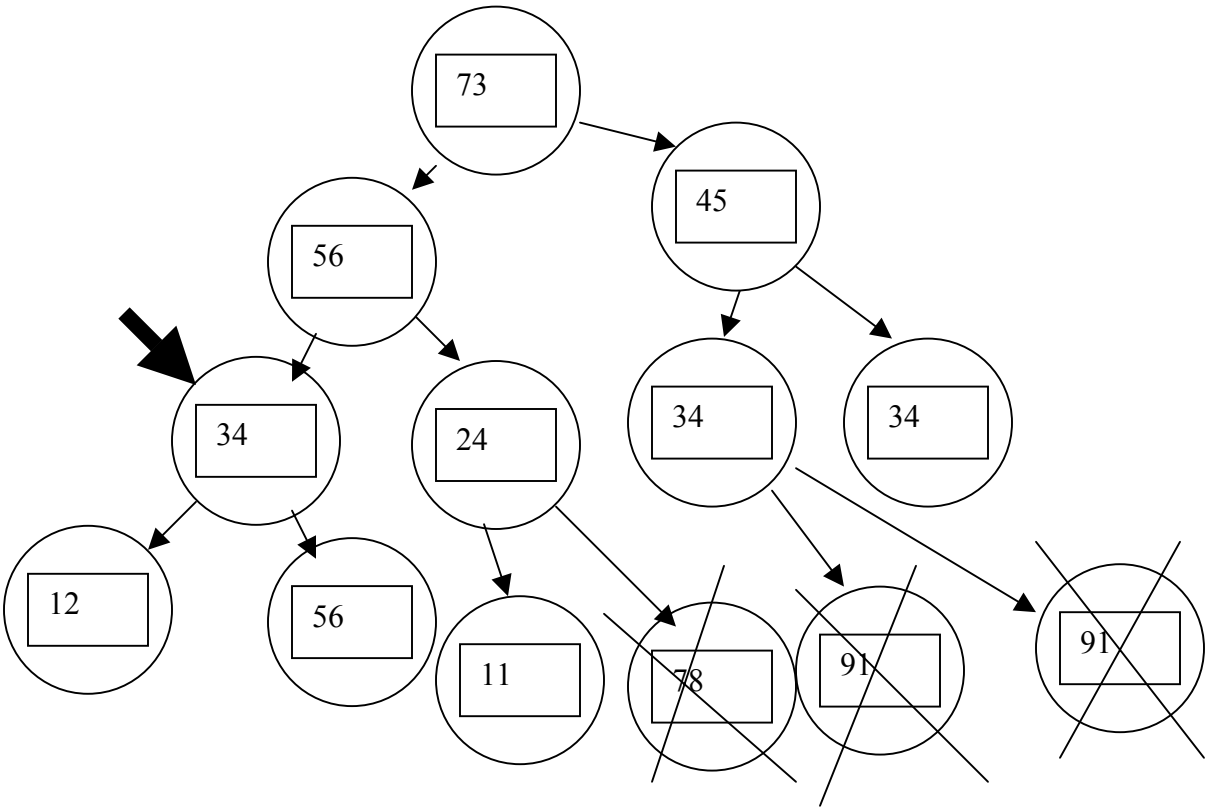
The root of the tree has 34, which is compared with its two children, and the greater 73 goes up and 34 comes down.

1	2	3	4	5	6	7	8	9	10	11	12	13
73	34	45	56	24	34	34	12	56	11	78	91	91



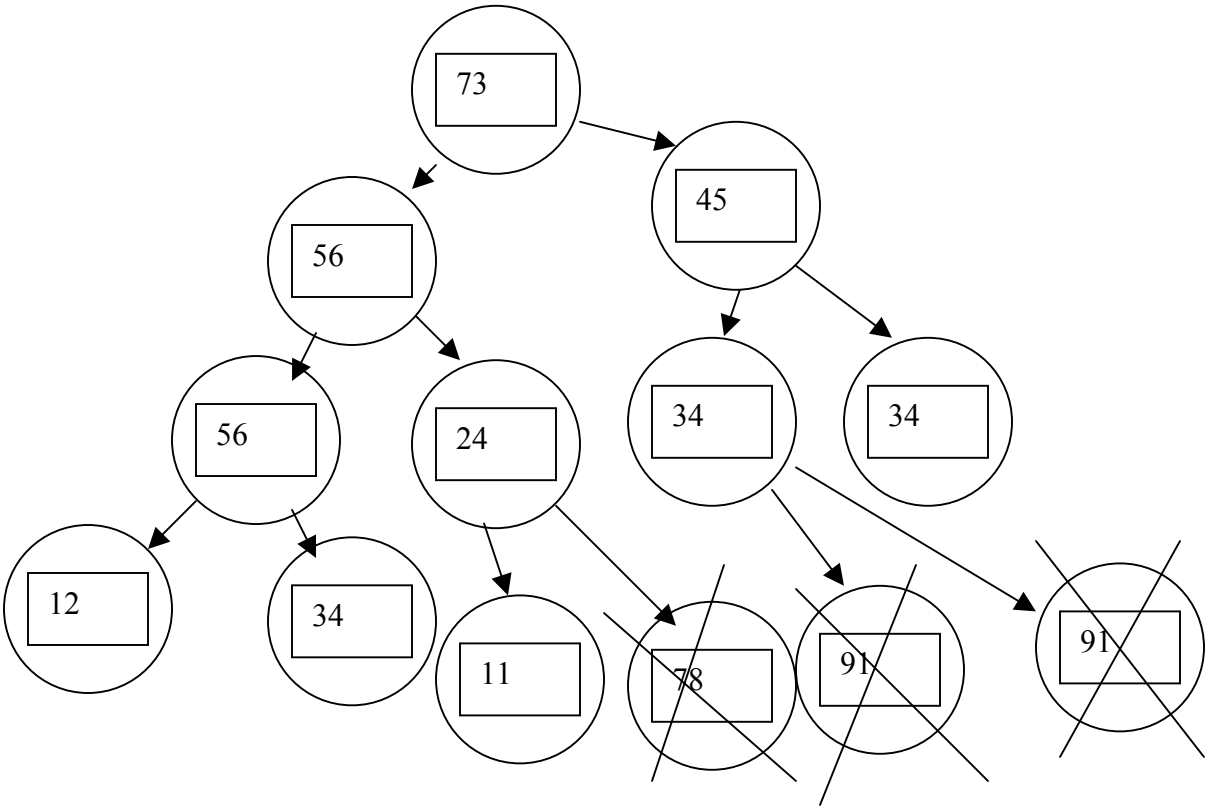
34 is now compared with its two children and 56 the greater one moves up and 34 moves down.

1	2	3	4	5	6	7	8	9	10	11	12	13
73	56	45	34	24	34	34	12	56	11	78	91	91



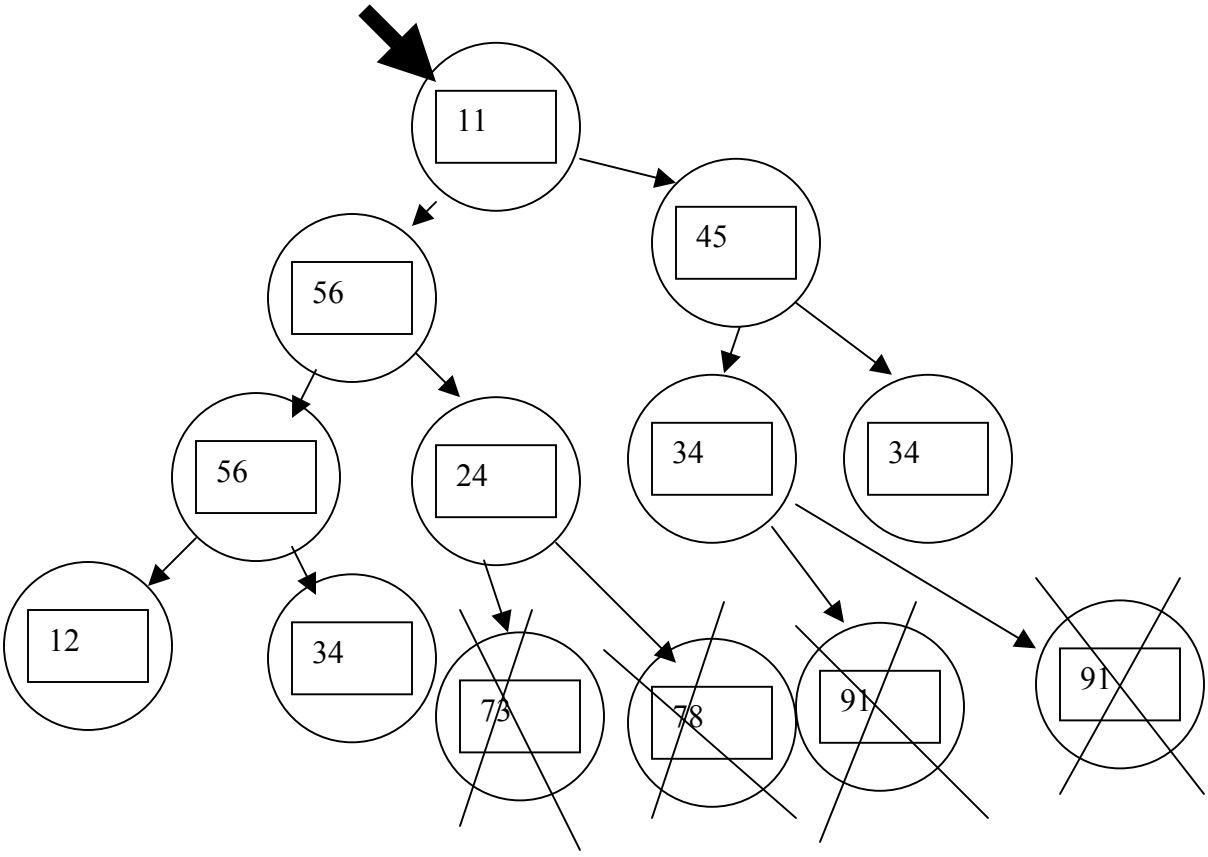
Now 34 is compared with its two children and the greater 56 moves up and 34 moves down, giving a heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
73	56	45	56	24	34	34	12	34	11	78	91	91



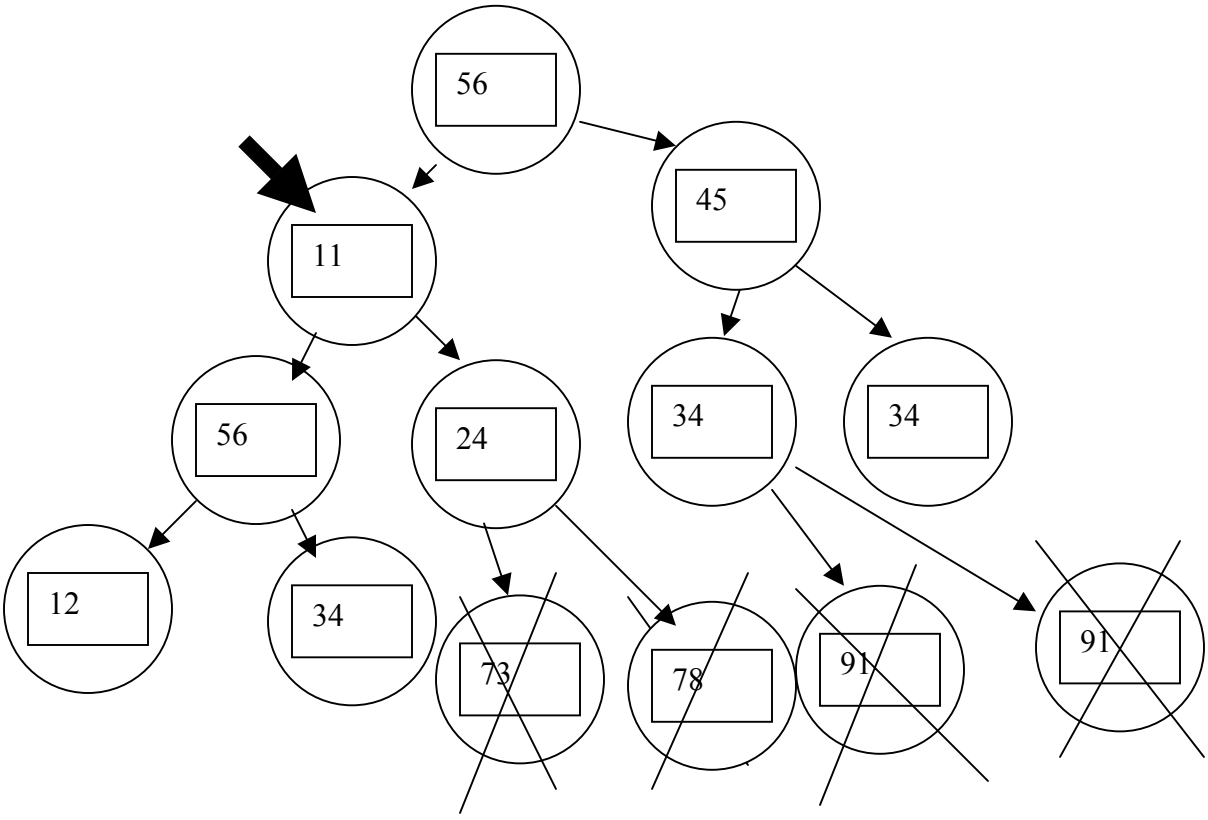
Now 73 at the root of the heap is sent to the bottom of the heap and we ignore the bottom of the heap therafter.

1	2	3	4	5	6	7	8	9	10	11	12	13
11	56	45	56	24	34	34	12	34	73	78	91	91



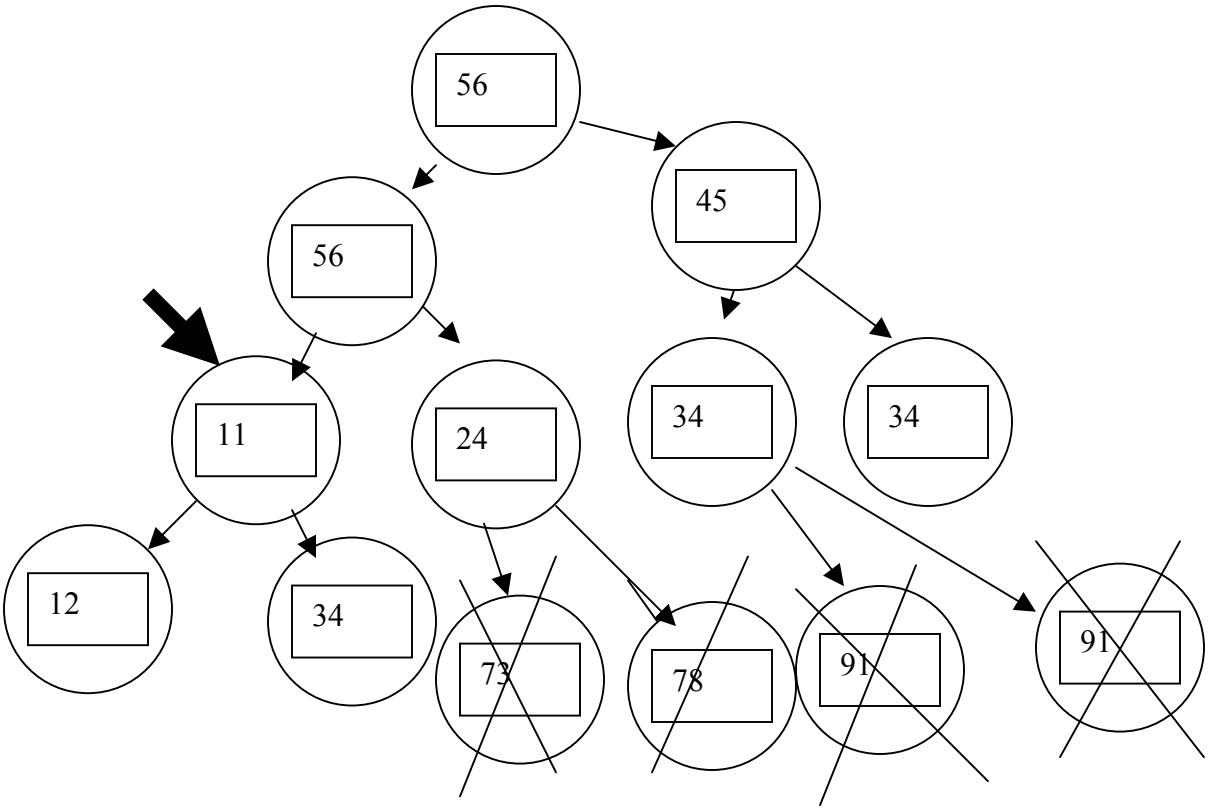
Now 11 has to be properly adjusted, it is replaced by the greater of its two children i.e.56. 11 moves down to 56's position.

1	2	3	4	5	6	7	8	9	10	11	12	13
56	11	45	56	24	34	34	12	34	73	78	91	91



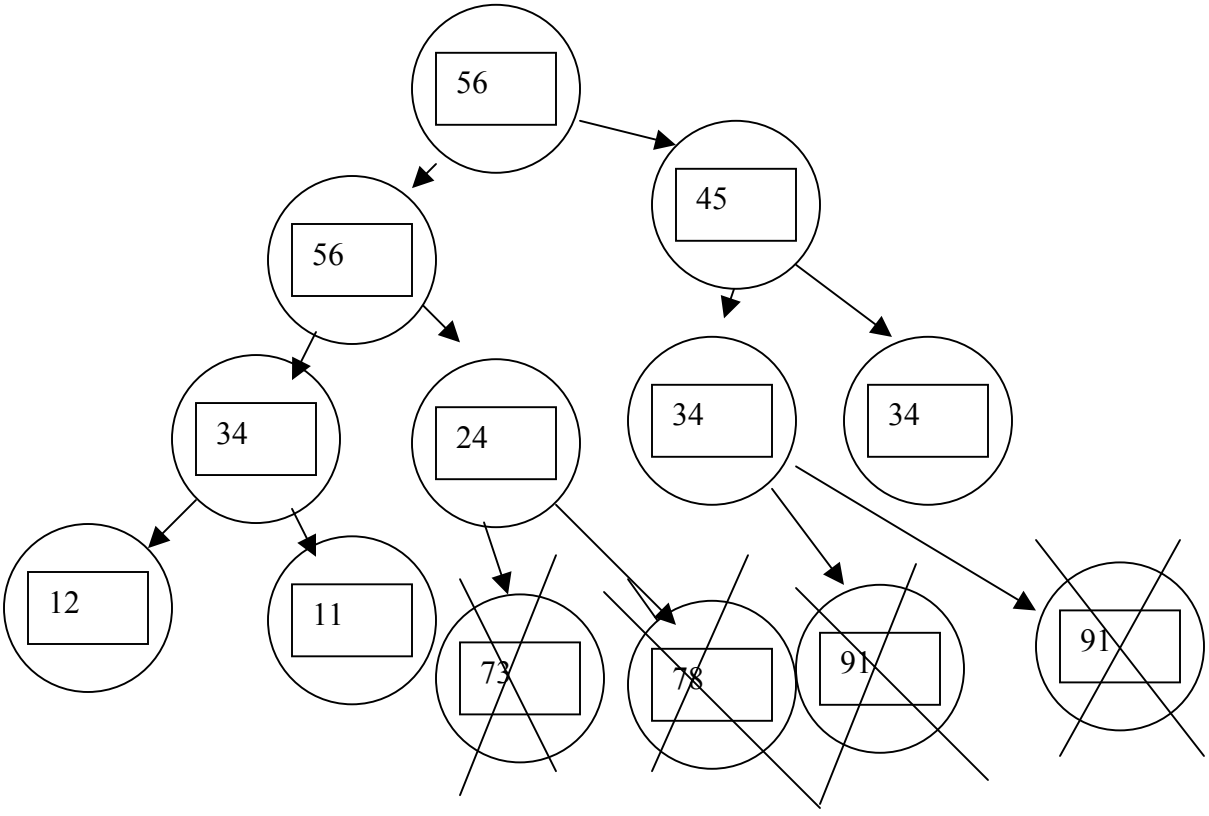
11 goes down to the position of its larger child at 56.

1	2	3	4	5	6	7	8	9	10	11	12	13
56	56	45	11	24	34	34	12	34	73	78	91	91



11 now finally moves down to the position of its larger child 34.

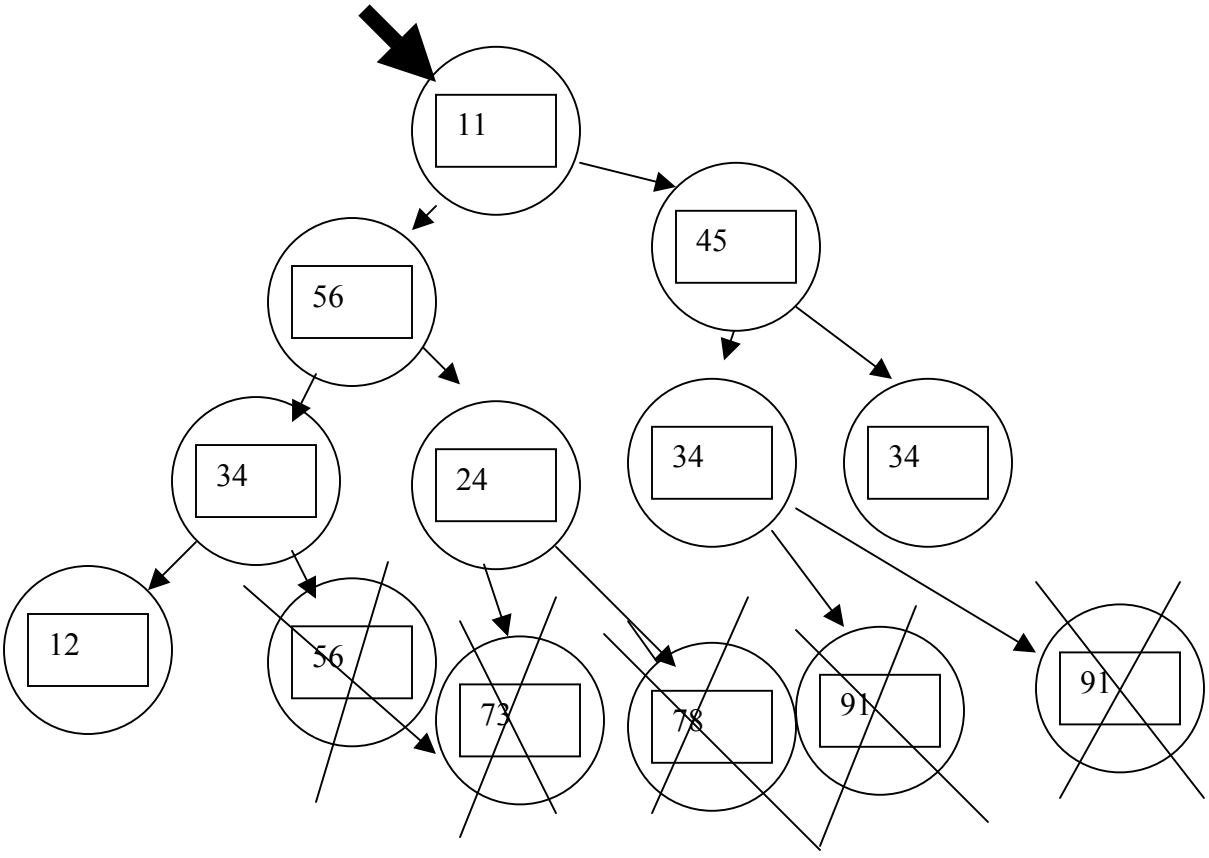
1	2	3	4	5	6	7	8	9	10	11	12	13
56	56	45	34	24	34	34	12	11	73	78	91	91



Above is a heap.

Now delete 56 from the heap, by swapping it with the element at the bottom of the heap.

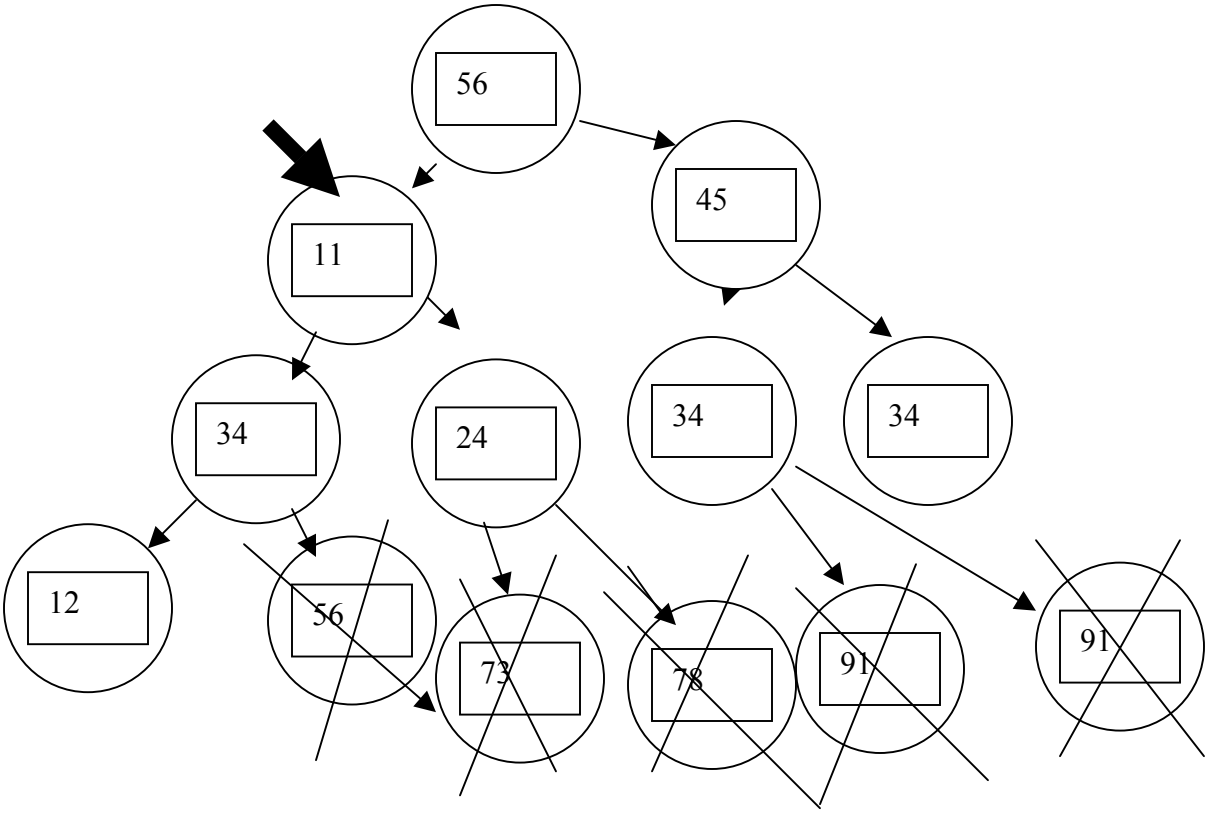
1	2	3	4	5	6	7	8	9	10	11	12	13
11	56	45	34	24	34	34	12	56	73	78	91	91



Now 11 is violating the heap property and it has to be adjusted.

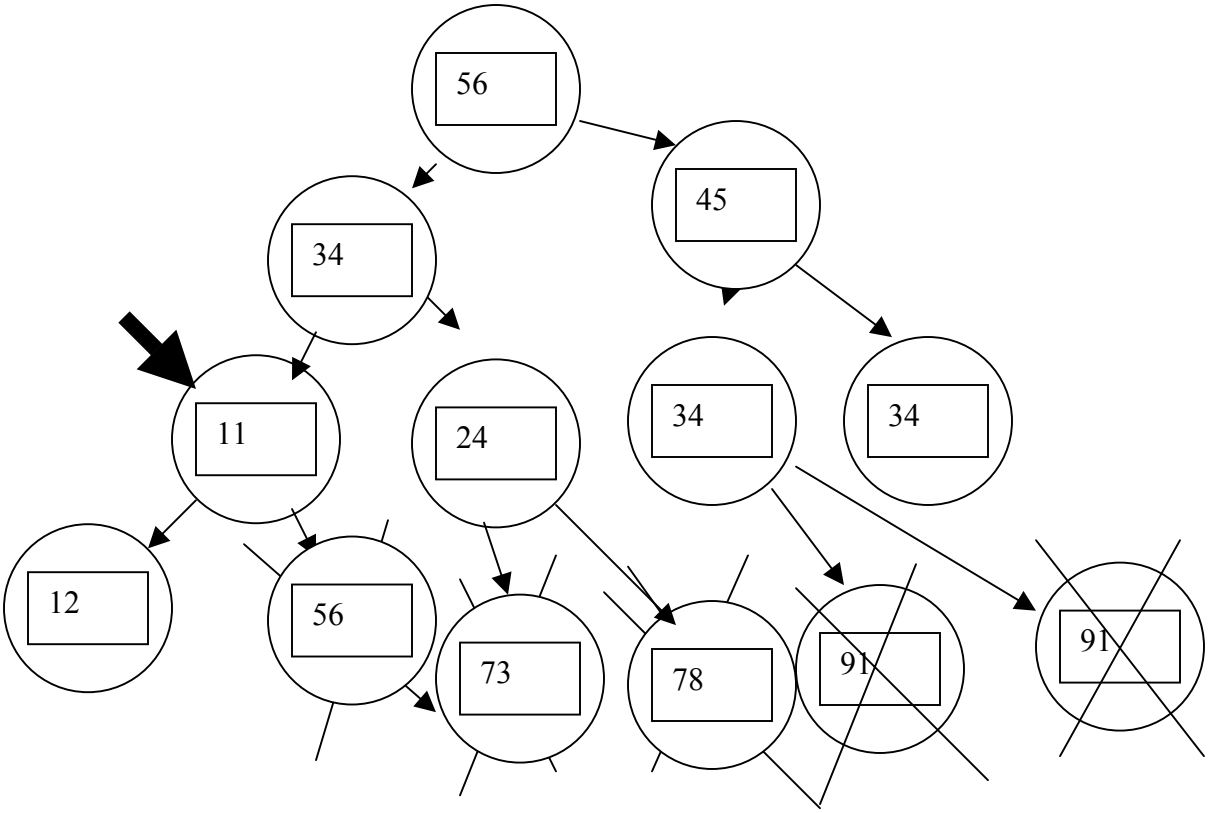
Compare 11 with its two children. 56 is larger, so it is swapped with 11.

1	2	3	4	5	6	7	8	9	10	11	12	13
56	11	45	34	24	34	34	12	56	73	78	91	91



Compare 11 with its two children. 34 is larger, so it is swapped with 11.

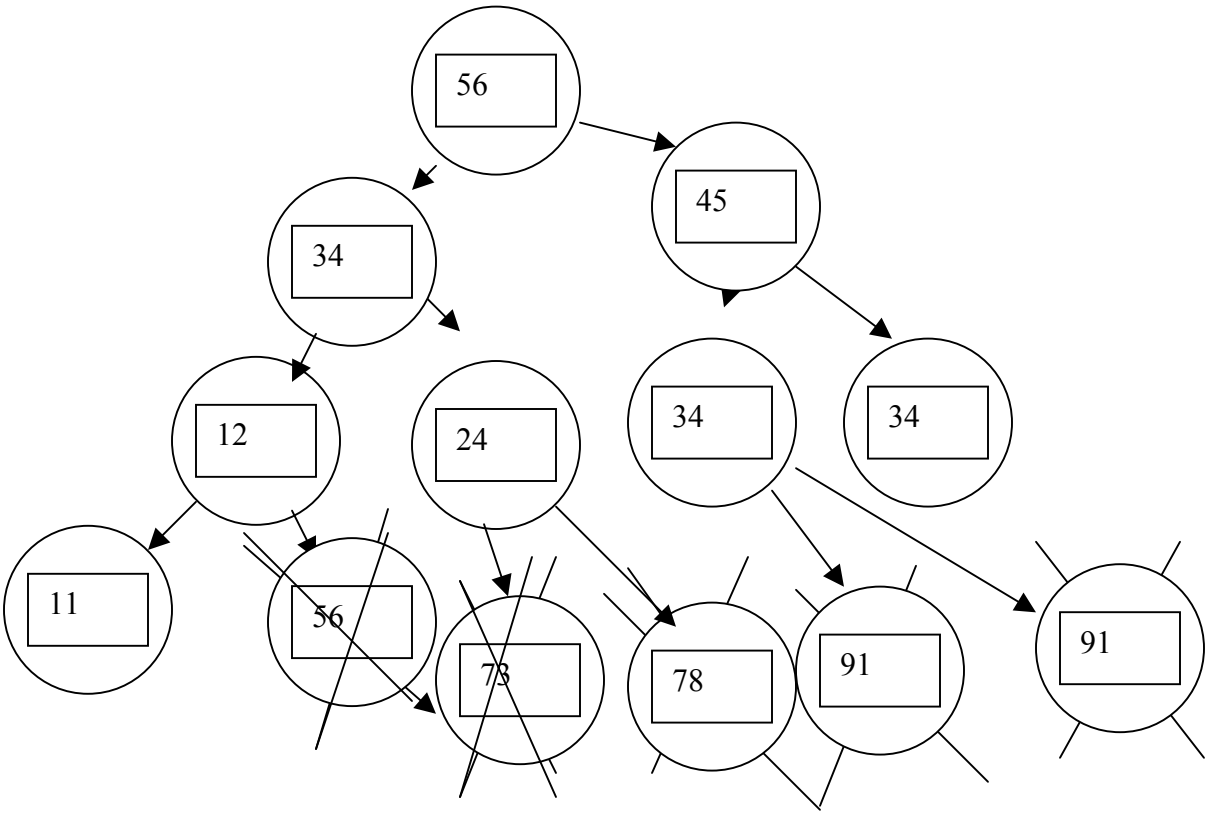
1	2	3	4	5	6	7	8	9	10	11	12	13
56	34	45	11	24	34	34	12	56	73	78	91	91



11 has one child 12 which is greater, so it is swapped with it.

Compare 11 with its two children. 34 is larger, so it is swapped with 11.

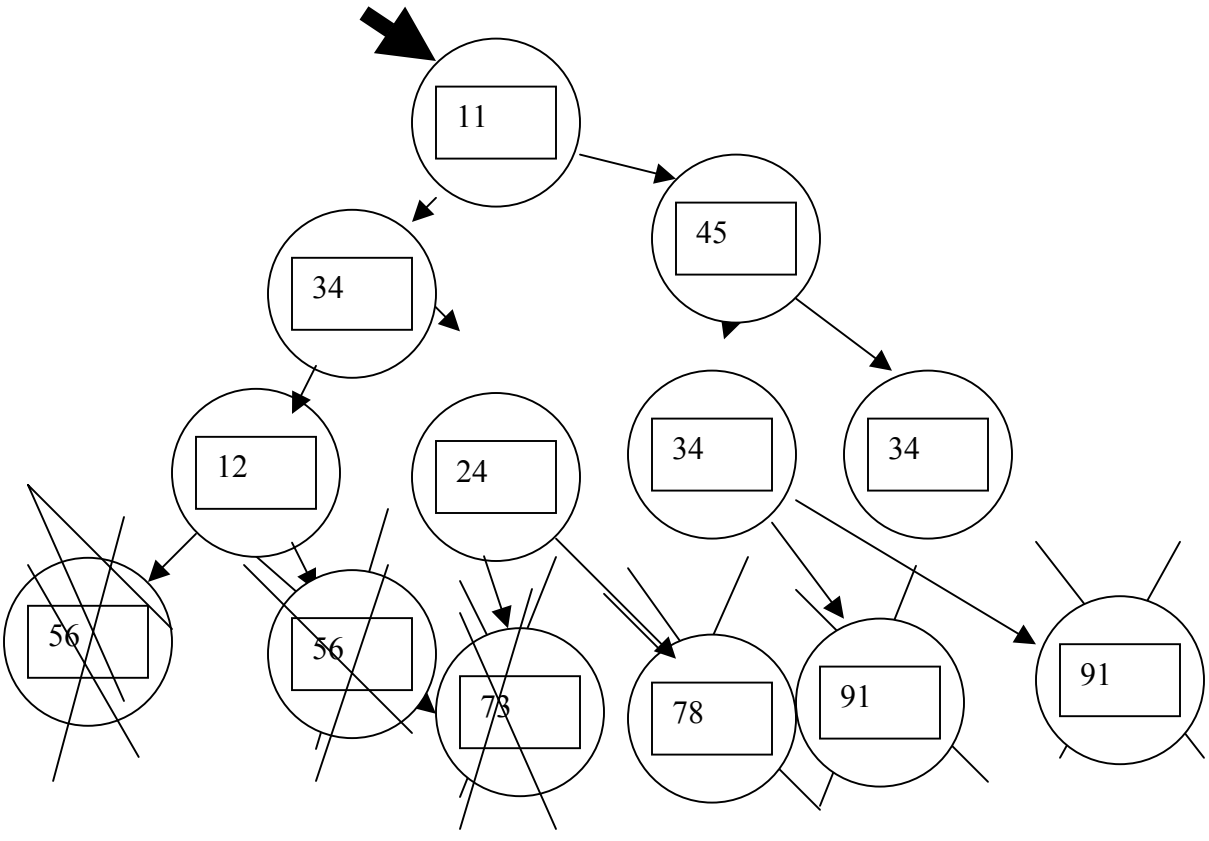
1	2	3	4	5	6	7	8	9	10	11	12	13
56	34	45	12	24	34	34	11	56	73	78	91	91



ABOVE IS A HEAP.

Now we delete 56, the maximum element in the heap by interchanging it with the bottom of the heap.

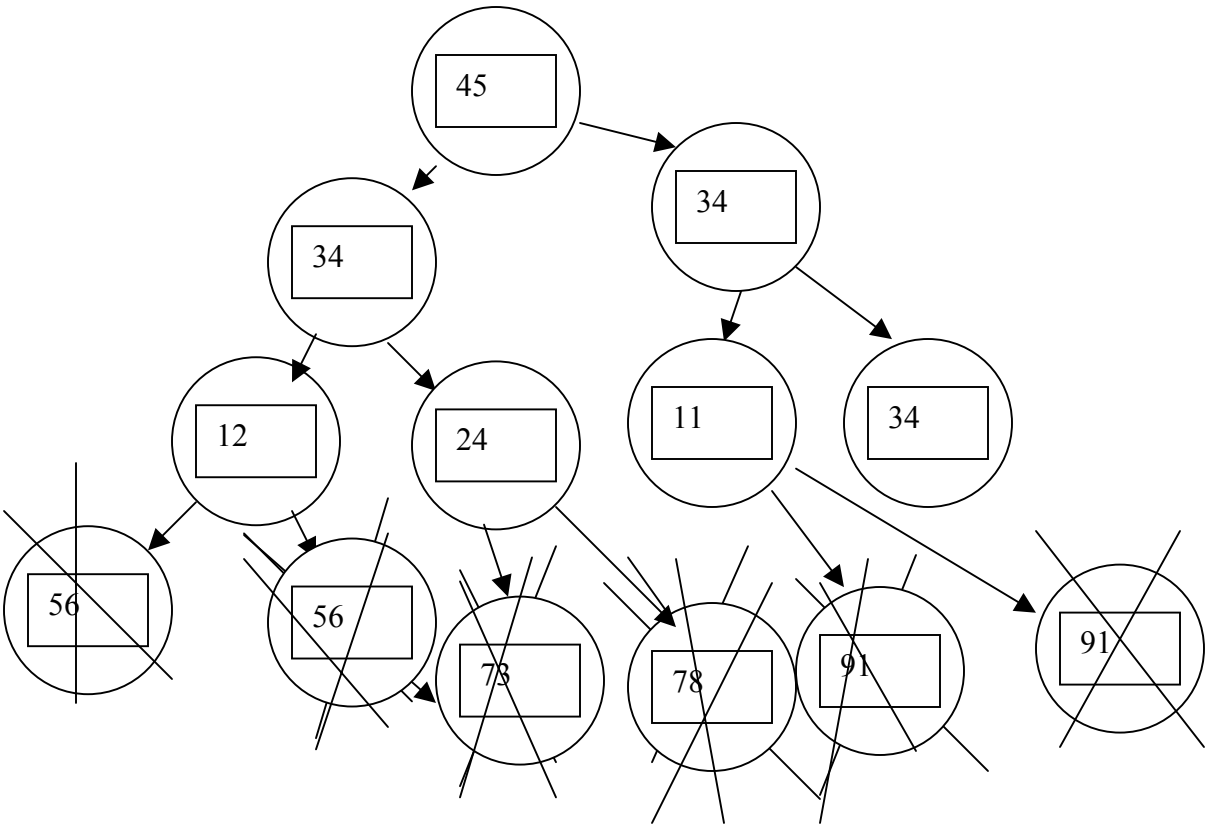
1	2	3	4	5	6	7	8	9	10	11	12	13
11	34	45	12	24	34	34	56	56	73	78	91	91



11 has to be put in its proper position.

11 is compared with its two children and 45 goes up and 11 comes down.

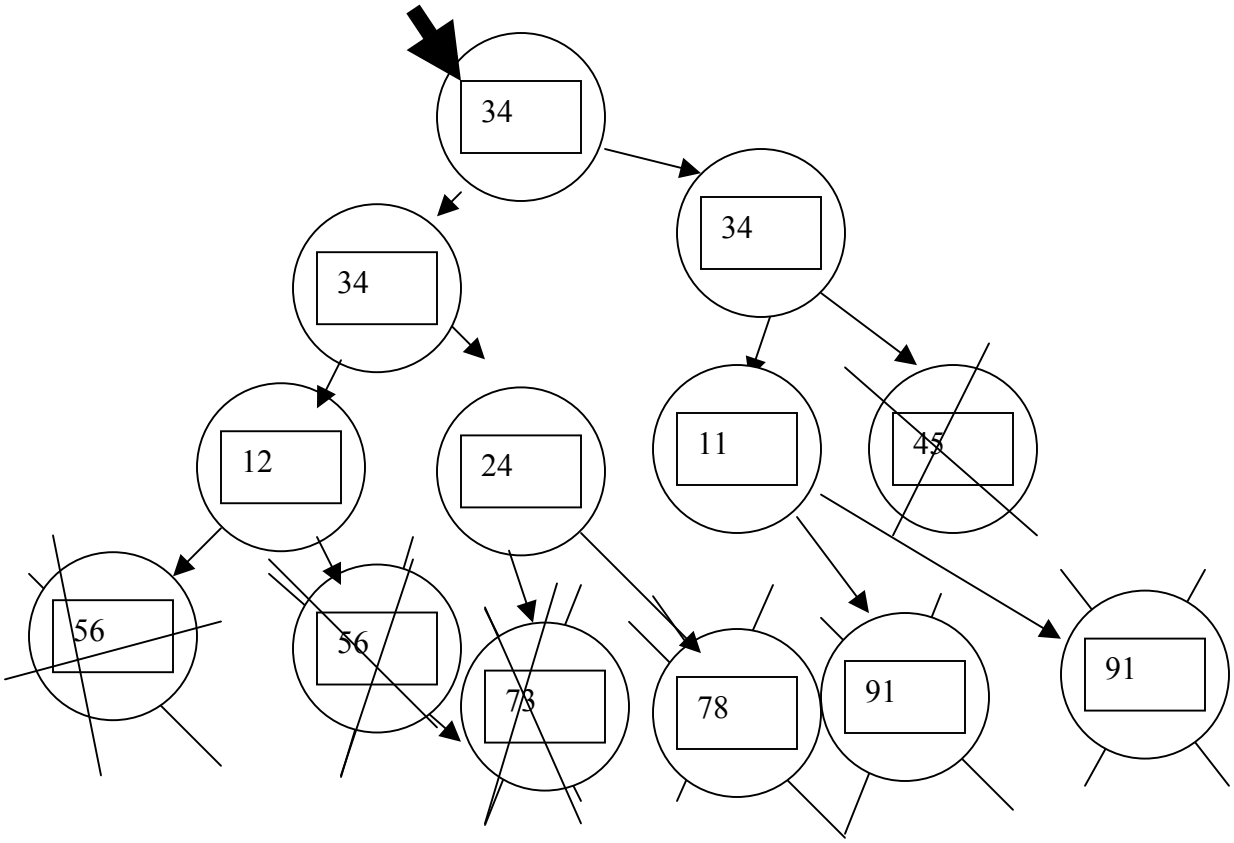
1	2	3	4	5	6	7	8	9	10	11	12	13
45	34	11	12	24	34	34	56	56	73	78	91	91



ABOVE IS A HEAP.

Now 45 is deleted form the heap by swapping it with the element at the bottom of the heap.

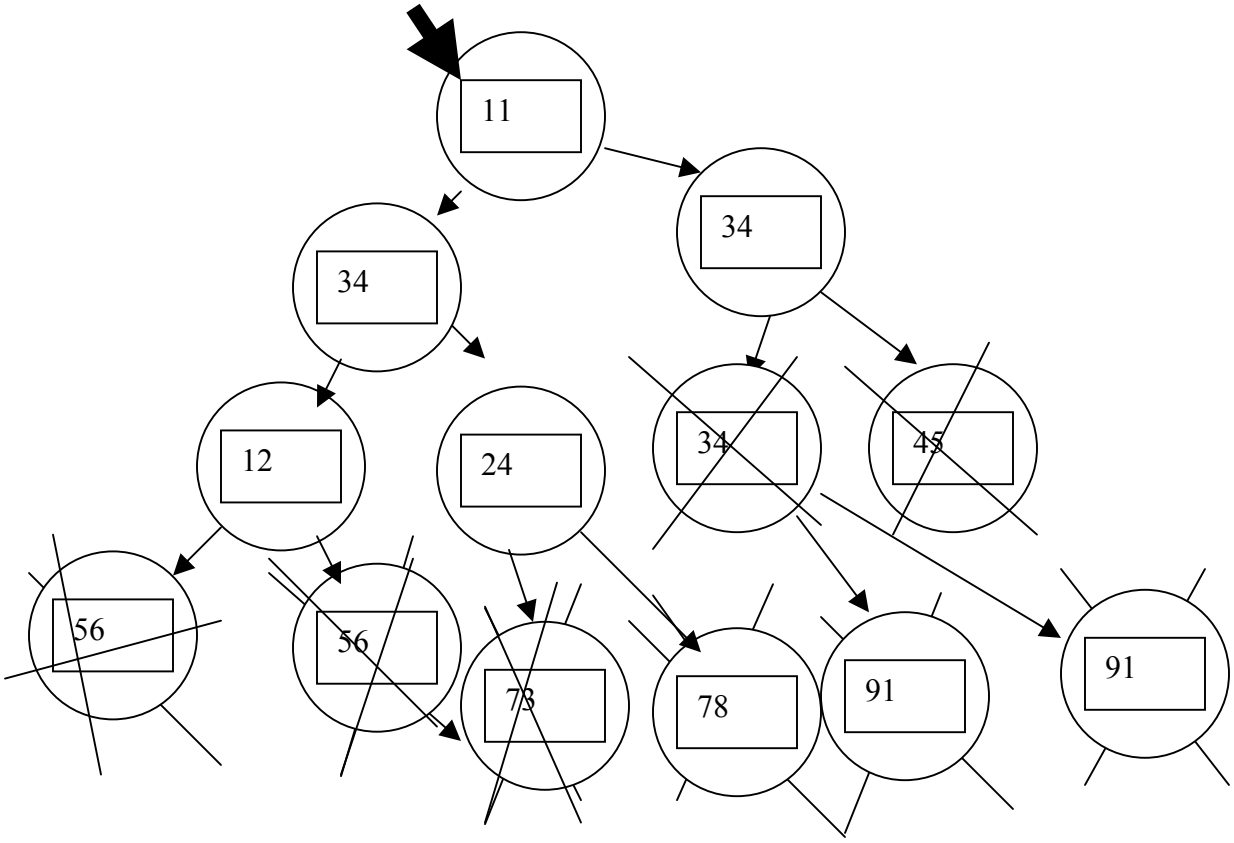
1	2	3	4	5	6	7	8	9	10	11	12	13
34	34	34	12	24	11	45	56	56	73	78	91	91



Above is a heap.

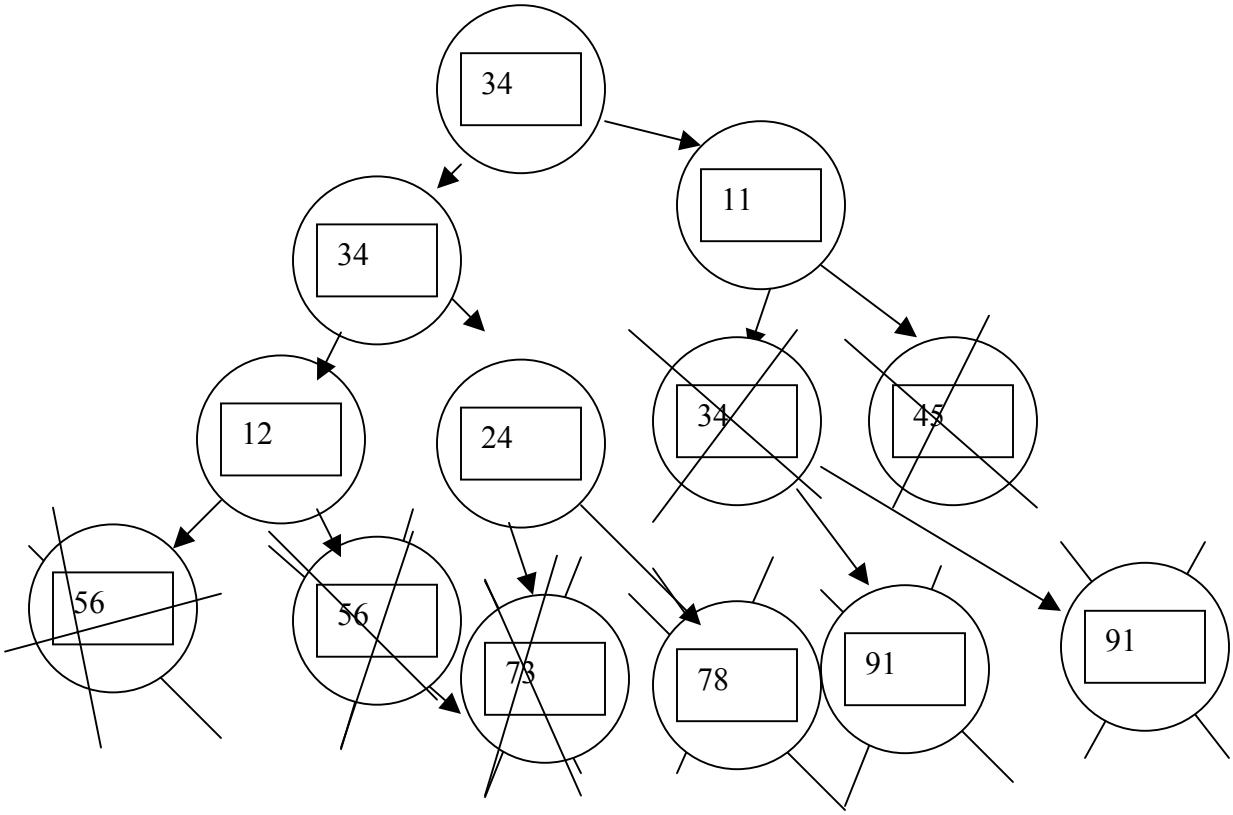
Now 34 at the root of the heap is sent to the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
11	34	34	12	24	34	45	56	56	73	78	91	91



The heap has to be adjusted, so we swap the root 11 with the 34 at the right child.

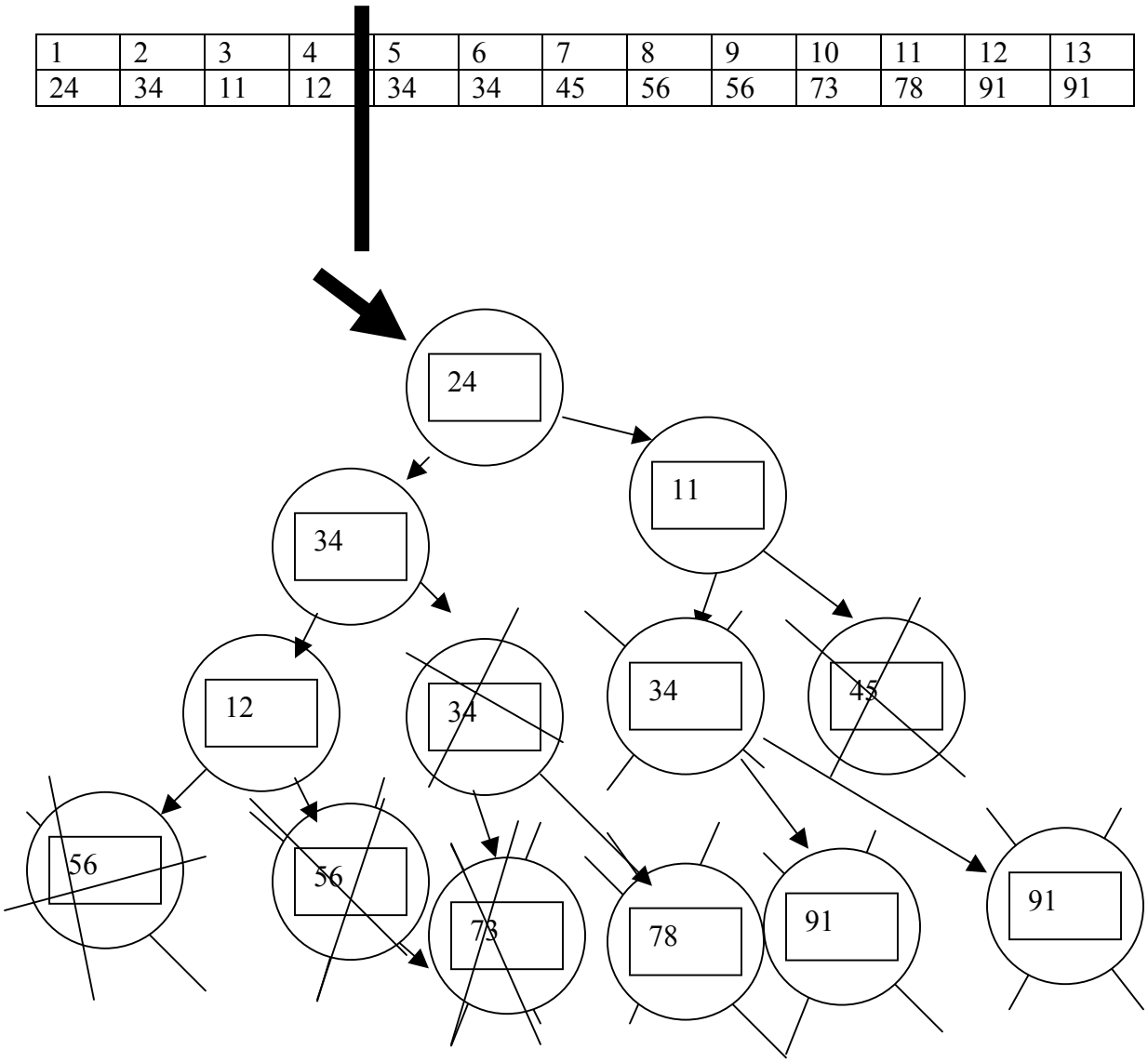
1	2	3	4	5	6	7	8	9	10	11	12	13
34	34	11	12	24	34	45	56	56	73	78	91	91



ABOVE IS A HEAP.

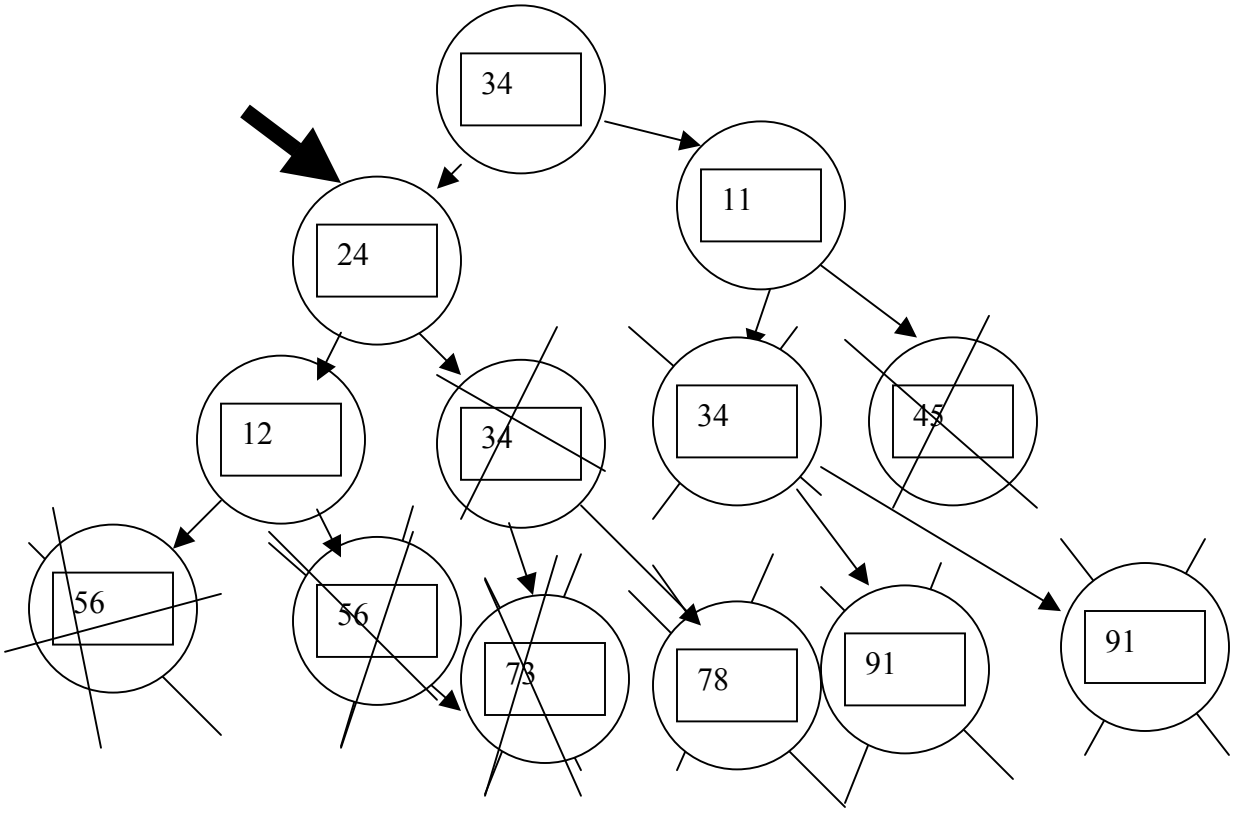
Now 34 the maximum element at the root is deleted from the heap by sending it to the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
24	34	11	12	34	34	45	56	56	73	78	91	91



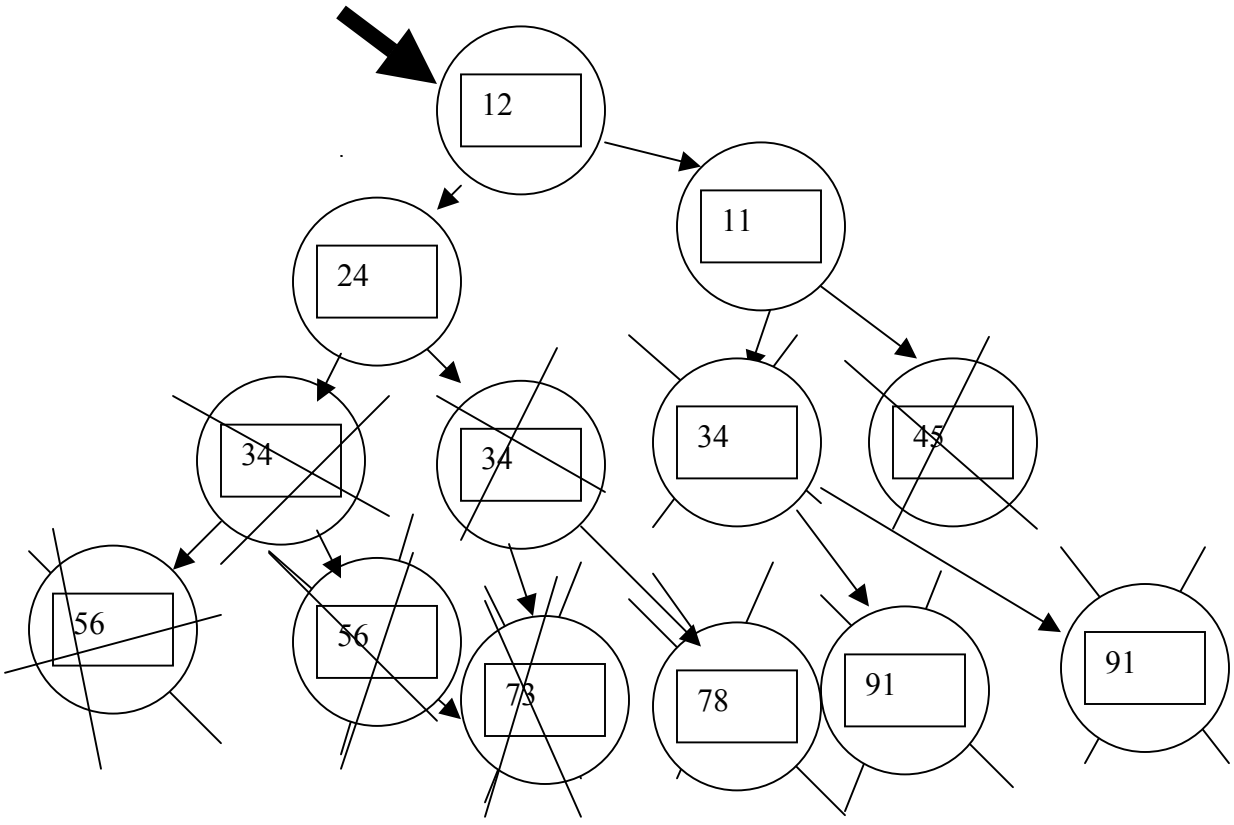
Now 24 is violating the heap property. So it is replaced by the greater of its two children i.e. the left child which is 34.

1	2	3	4	5	6	7	8	9	10	11	12	13
34	24	11	12	34	34	45	56	56	73	78	91	91



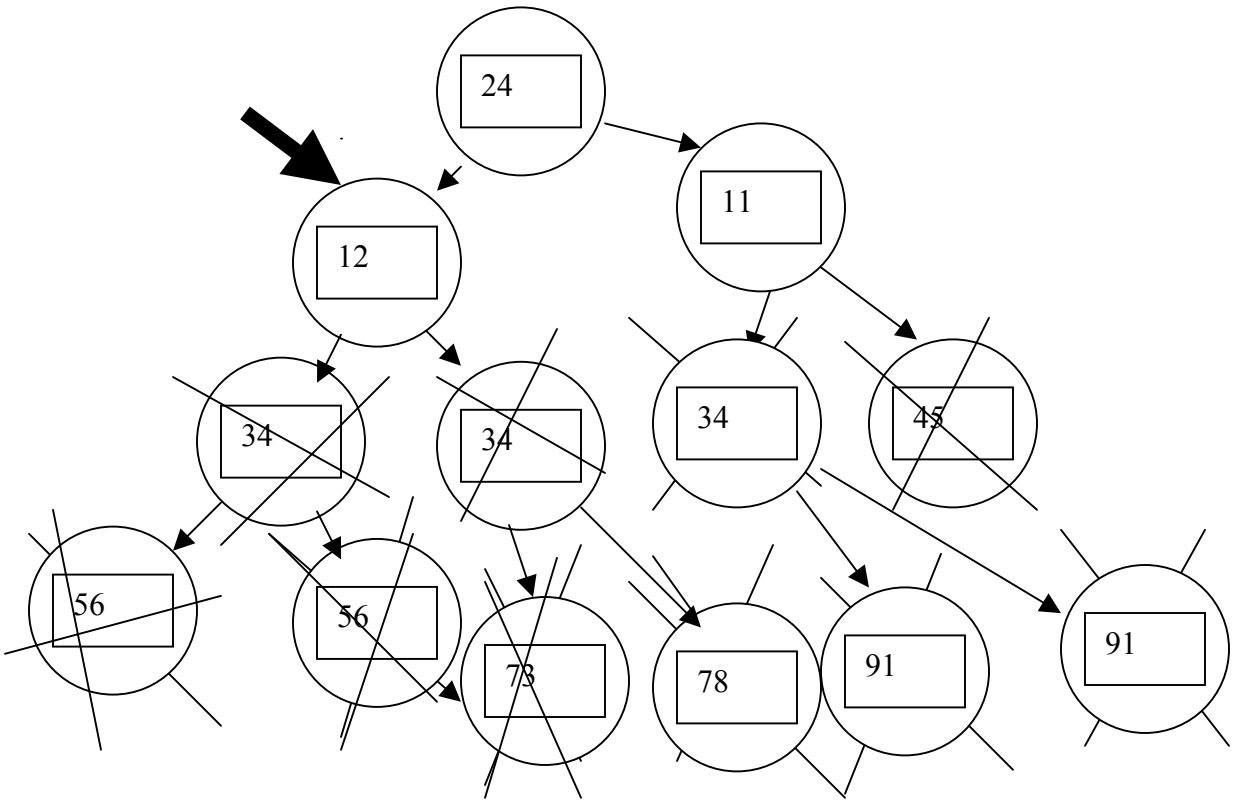
Now 34 is deleted from the heap and sent to the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
12	24	11	34	34	34	45	56	56	73	78	91	91



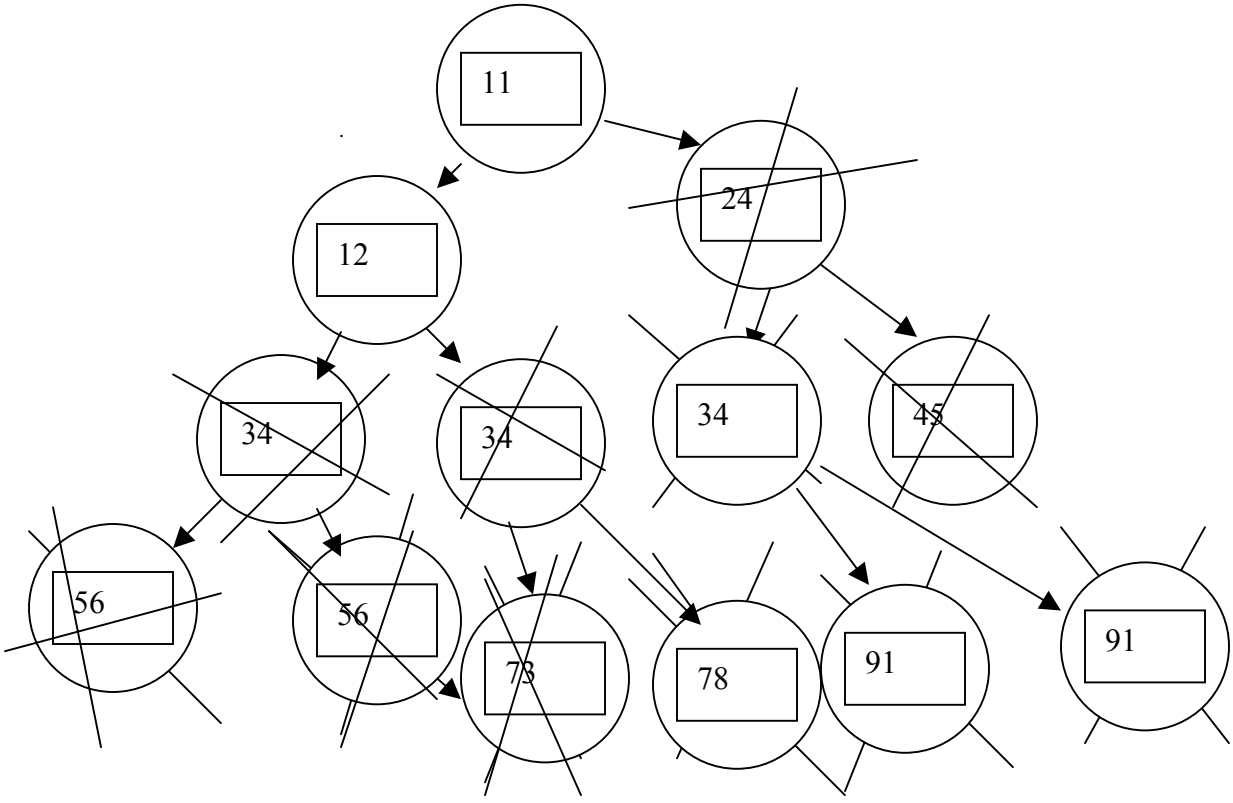
Now 12 is violating the heap property so it is replaced by its larger child 24.

1	2	3	4	5	6	7	8	9	10	11	12	13
24	12	11	34	34	34	45	56	56	73	78	91	91



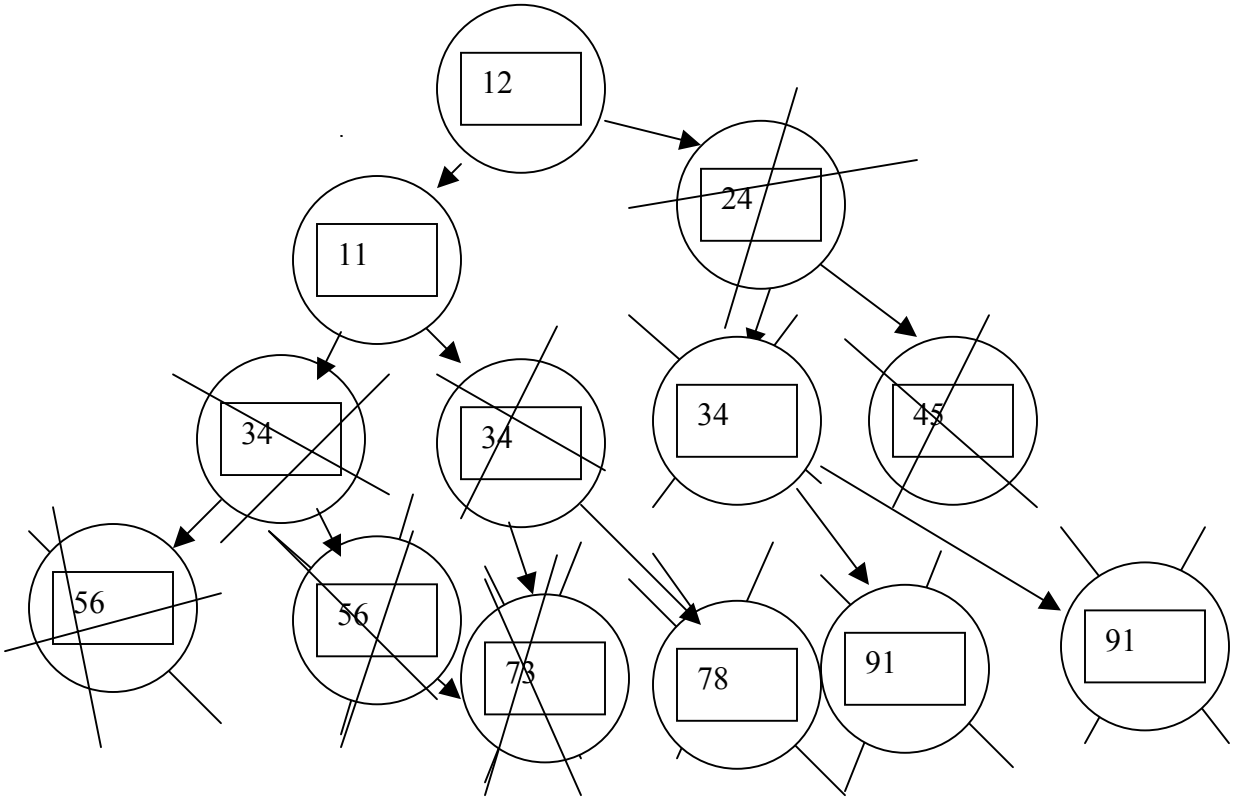
24 is swapped with the element at the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
11	12	24	34	34	34	45	56	56	73	78	91	91



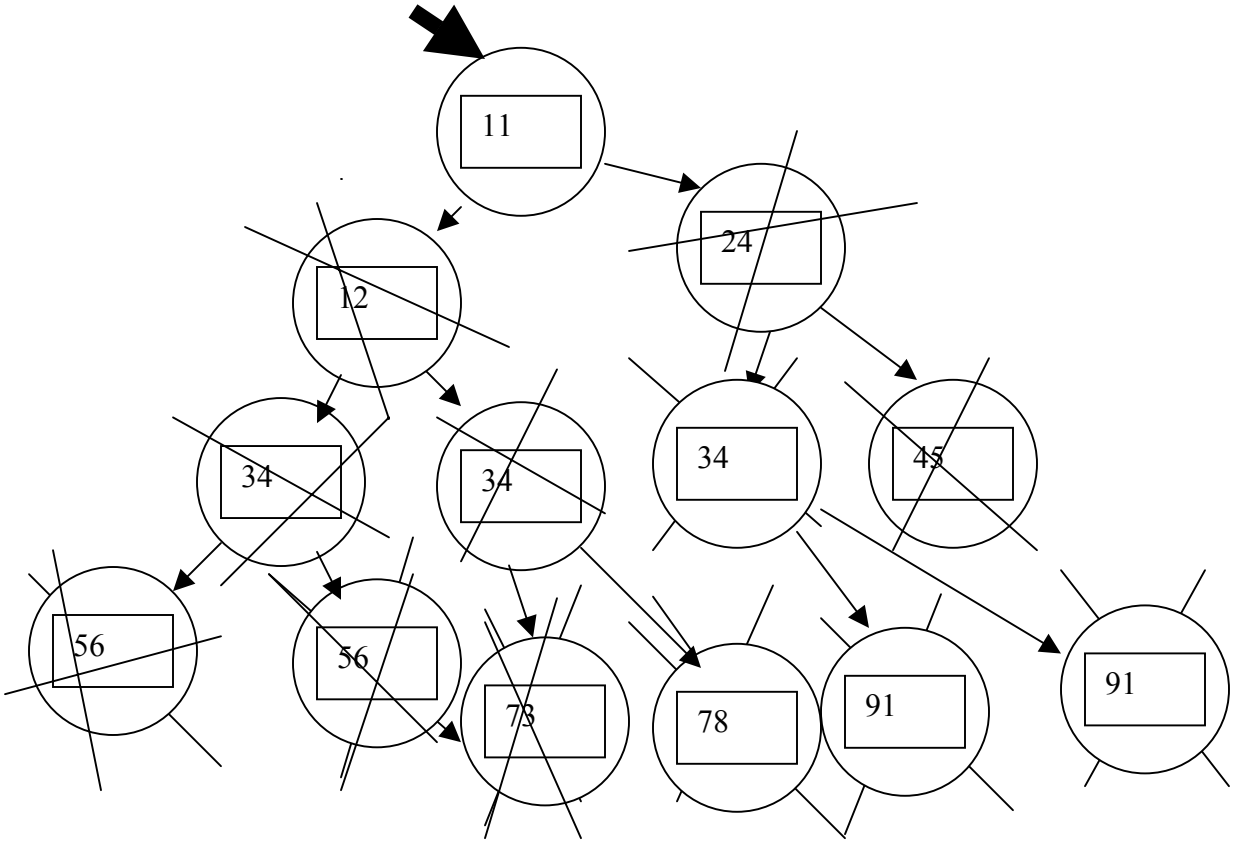
11 is violating the heap property. So it is swapped by with its large child 12.

1	2	3	4	5	6	7	8	9	10	11	12	13
12	11	24	34	34	34	45	56	56	73	78	91	91



Interchanging 12 with the bottom of the heap.

1	2	3	4	5	6	7	8	9	10	11	12	13
11	12	24	34	34	34	45	56	56	73	78	91	91



Above is a heap.

Now only one element is left in the heap.
So we have a sorted list

1	2	3	4	5	6	7	8	9	10	11	12	13
11	12	24	34	34	34	45	56	56	73	78	91	91

END OF SIMULATION