1) (3 marks) Which of the following trees are heaps? Explain your answers.

- Tree (i) is not a heap because it is not a complete binary tree.
- Tree (ii) is a max-heap because it is a complete binary tree and for each internal node in this tree, the value stored at that node is greater than the value stored at any of its descendants.
- Tree (iii) is a min-heap because it is a complete binary tree and for each internal node in this tree, the value stored at that node is less than the value stored at any of its descendants.

2) (3 marks) Show the addition of 7 to the following min-heap:

3) (3 marks) Show the deletion of the smallest value from the following min-heap:
4) (3 marks) Show the addition of 32 to the following max-heap:

```
33
  /  \
12   29
  / /  \
10 9 4 1 3
```

5. (8 marks) Given a max-heap Java class `IntMaxHeap` with existing methods

```java
public IntMaxHeap()
public int size()
public void add(int val)
public int getMax(int val)
```

where `add` only accepts values that are greater than zero and `getMax` deletes the largest value stored in the heap and returns that value, write the body of a new method for this class,

```java
public int nth(int n)
```

which returns the nth smallest value in the heap if $0 < n \leq size$ and $-1$ otherwise. Note that this method must leave the set of elements stored in the heap unchanged (though it can change this set during the operation of the method). For example, given the max-heap in Question (4) above, the values returned by this method when $n = 1, 3, 7, 9, \text{ and } 14$ are $1, 3, 12, 33, \text{ and } -1$, respectively.

```java
public int nth(int n){
    int[] temp;
    int i, num_getMax;

    if (!((0 < n) && (n <= size())))
        return(-1);

    num_getMax = (size() - n) + 1;
    temp = new int[num_getMax];
    for (i = 0; i < num_getMax; i++)
        temp[i] = getMax();
    for (i = 0; i < num_getMax; i++)
        add(temp[i]);
    return(temp[num_getMax - 1]);
}
```