1. (a) Draw the binary tree produced by inserting the following elements in a max-heap in the given order:

3, 5, 2, 1, 6, 8, 7

(Show your work.)

(b) List the elements of the array which represents the heap in problem 1a. List the elements in the order they appear in the array.

2. Consider the following max heap:

(a) Apply the ExtractMax() operation to the heap in Figure 1 and draw the resulting binary tree. (Show your work.)

(b) Apply the operation Insert(64) to the heap in Figure 1 and draw the resulting binary tree. (Apply Insert(64) to the heap exactly as given in Figure 1. Do not apply the ExtractMax() in step (a) before applying Insert(64). Show your work.)

3. Consider the following program:

```plaintext
Program1(A, n)
   /* A is an array of n elements */
1    P.Initialize();
2    for i ← 1 to ⌊√n⌋ do
3       for j ← 1 to n do
5       end
6    end
7    while P.Size() ≠ 0 do
8       x ← P.ExtractMax();
9       Print x;
10   end
```

P.Initialize() initializes the data structures.
P.Insert(x) inserts elements x in P.
P.ExtractMax() returns the maximum element of P and deletes it from P.
P.Size() returns the number of elements in P.

Analyze carefully the running time of Program1 assuming that P is implemented as a Max-Heap. Note that the time for P.Insert() and P.ExtractMax() is dependent on the number of elements in P which changes over the running time of the algorithm. Operations P.Initialize() and P.Size() take constant time.

Show your work.
4. Consider the following program:

```plaintext
Program2(A, n)
    /* A is an array of n elements */
    1 P.Initialize();
    2 for i ← 1 to ⌊√n⌋ do
        3     for j ← 1 to n do
        end
        5     x ← P.ExtractMax();
    end
    6 Print x;
end
```

P.Initialize() initializes the data structures.
P.Insert(x) inserts elements x in P.
P.ExtractMax() returns the maximum element of P and deletes it from P.
P.Size() returns the number of elements in P.

(a) Analyze carefully the running time of Program2 assuming that P is implemented as a Max-Heap. Note that the time for P.Insert() and P.ExtractMax() is dependent on the number of elements in P which changes over the running time of the algorithm. Operations P.Initialize() and P.Size() take constant time.
   Show your work.

(b) Analyze carefully the running time of Program2 assuming that P is implemented by some data structure which takes Θ(s) time for P.Insert() where s is the number of elements in P and Θ(1) time for P.ExtractMax(). Note that the time for P.Insert() is dependent on the number of elements in P. Operations P.Initialize() and P.Size() take constant time.
   Show your work.

(c) Analyze carefully the running time of Program2 assuming that P is implemented by some data structure which takes Θ(1) time for P.Insert() and Θ(s) time for P.ExtractMax() where s is the number of elements in P. Note that the time for P.ExtractMax() is dependent on the number of elements in P. Operations P.Initialize() and P.Size() take constant time.
   Show your work.