1: Maximum: n. Each node is the child of the previous node, therefore the tree is like a long rope with the height of n.
Minimum: lg(n+1) -1. Each node has two children, therefore the n level has the formula 2^{h+1}-1, calculate for h the answer is lg(n+1)-1.

2: Maximum: n. Each node is the child of the previous node, so the height is n.
Minimum: 1. Each node is the child of the root, therefore the height is 1.

4.6: MAKENULL = DELETE < MEMBER < MIN < INSERT < UNION < INTERSECTION
a < b < d < c

4.7:

open hash table:

0	343(343 mod 7 = 0)				
1	1(1 mod 7 = 1)	--->	8(8 mod 7 = 1)	--->	64(64 mod 7 = 1)
2					
3					
4					
5					
6	27(27 mod 7 = 6)	--->	125(125 mod 7 = 6)	--->	216(216 mod 7 = 6)

closed hash table:

0-8(8 mod 7 = 1)				
1-1(1 mod 7 = 1)				
2-64(64 mod 7 = 1)				
3-125(125 mod 7 = 6)				
4-216(216 mod 7 = 6)				
5-343(343 mod 7 = 0)				
6-27(27 mod 7 = 6)				

5: 1. The value of h1(x) should belong to the class of x. Therefore, the value of h1(x) should be string, not the integer(length of string).
2. The random number may be repeated. If h2(x) already exist the number, the collision occurs. And if the table is full, then nothing will be inserted to the table, and the return r will be null or error.

6:
```
procedure DELETE(S: SET, i: integer);
begin:
    for t:=1 to n do
        if S[t] = i do
            S[t] = null
        fi
    end
end
```
complexity: O(n)

```
procedure ADD(S: SET, i: integer);
begin:
    for t:=1 to n do
        if S[t] == i do
            return false
        else
            S[n+1] = i
        fi
    end
end
```
complexity: O(n)

7:
```
procedure INCREASE(h0: HASH TABLE)
h1:= double size of h0
begin:
    for h in h0 do
        h0(h) = h1(h)
    end
end
```
complexity: O(n)