Question 1

```
//bubble sort
Algorithm beautiful(A, n)
        for int i <- 0 to n do
                for int j <- 0 to n-i do
                        temp <- arr[j]
                arr[j] <- arr[j+1]
                arr[j+1] <- temp
```

In every case, in bubble sort, we still have to traverse through all the elements twice for a loop. So that is n^2 complexity for best and worst-case same.

Question 2

ascending: 2^n, 2^(n + 1) 2^(2n), 2^(2^n)

Question 3

O(1) - add element to array
O(log n) - binary search
O(n) - find min element
O(n log n) - merge sort
O(n^2) - bubble sort
O(n^3) - 3 variables equation solver
O(2^n) - Find all subsets

Question 4

We can't use the master theorem for Fibonacci n. So in this case we can use Counting self-calls:
S(0) = 0, S(1) = 0
So the self calls of Fibonacci n will be:
S(n) = 2 + S(n-1) + S(n-2)

Question 5

T(1) = 1
T(n) = 2T(n/2) + c
N = 2^k
Counting self-calls:
S(n^k) = 2 + S(n^k-1) + S(n^k-2)

Question 5-2

Master theorem examples:

1) T(n) = T + (2/2) + cn^2
a = 1 b = 2 c = 2

1 < 2^2: That means the time complexity will be: Θ(n^k)

2) T(n) = 2T + (n/2) + n
a = 2 b = 2 c = 2

2 < 2^2: That means the time complexity will be: Θ(n^k)

3) T(n) = 2T + (2/2) + n
 a = 2 b = 4, d = 1/2

2 = 4^(1/2): That means the time complexity will be: Θ(n^k log n)