Question 1-)

a-)

```
public List<Integer> findMaxLenSortedSublist(List<Integer>
list)
{
    int max = 1;
    int len = 1;
    int lastIndexOfSublist = 0;
    List<Integer> sublist = new ArrayList<>();

    for(int i = 1; i < list.size(); ++i)
    {
        if(list.get(i - 1) <= list.get(i))
            ++len;

        else
        {
            if(len > max)
            {
                max = len;
                lastIndexOfSublist = i;
            }
            len = 1;
        }
    }

    if(len > max)
    {
        max = len;
        lastIndexOfSublist = list.size();
    }
    for(int i = lastIndexOfSublist - max; i <
lastIndexOfSublist; ++i)
        sublist.add(list.get(i));

    return sublist;
}
```

Complexity Analysis-) Constant complexity for first for line + 4n complexity for first for loop(worst case) + constant complexity for if statement + n complexity for second for loop(worst case)

⇨ this method has T(n) = Θ(n) time complexity.

b-)

```
public List<Integer>
findRecursivelyMaxLenSortedSublist(List<Integer> list, int
index, int max, int len, int endingIndex)
{

    if(index == list.size()) //basis
    {
        List<Integer> sublist = new ArrayList<>();

        if(len > max)
        {
            max = len;
            endingIndex = list.size();
        }

        for(int i = endingIndex - max; i < endingIndex;
++i)
            sublist.add(list.get(i));

        max = 1; len = 1; endingIndex = 0;

        return sublist;
    }

    if(list.get(index-1) <= list.get(index))
        ++len;

    else
    {
        if(len > max)
        {
            max = len;
            endingIndex = index;
        }
        len = 1;
    }

    return findRecursivelyMaxLenSortedSublist(list,
++index, max, len,   endingIndex);//recursive call
}
```

Complexity Analysis-) Each recursive call we divide problem to two subproblem, and every sub problem has n-1 size => $T(n) = 2T(n-1) + 1$

Question 2-)

```java
static boolean hasPairWhoseSumIsX(int[] arr, int X)
{
    int l, r, size = arr.length;
    l = 0;
    r = size-1;
    while (l < r)
    {
        if(arr[l] + arr[r] == X)
            return true;
        else if(arr[l] + arr[r] < X)
            ++l;
        else
            --r;
    }
    return false;
}
```

hasPairWhoseSumIsX method has T(n) = Θ(n) time complexity in worst case scenario, worst scenario happens if there is no pair in sorted array or two middle elements are pair's component.

Question 3-)

```
for (i=2*n; i>=1; i=i-1)
    for (j=1; j<=i; j=j+1) ➔2n*(2n+1)/2 = n^2 + n
        for (k=1; k<=j; k=k*3) ➔ $\log_3 n$
            print( "hello" )
```

(n^2+n) ε Θ(n^2) => n^2 * $\log_3 n$) ε Θ(n^2logn)

Question 4-)

T(n) = 4T(n/2) + n^2

a = 4, b = 2, d = 2

a = 2^2 => n^d*logn ==> Θ(n^2logn)