Question - 1

a) Algorithm alg1(L[0...n-1]) $\rightarrow T(n)$

    if (n==1) return L[0] $\rightarrow 1$

    else

        tmp = alg1(L[0...n-2]) $\rightarrow T(n-1)$

        if (tmp <= L[n-1]) return tmp $\rightarrow 1$

        else return L[n-1] $\rightarrow 1$

$T(n) = T(n-1) + c$

$T(n) = (T(n-2)+c) + c$

$T(n) = ((T(n-3)+c)+c) + c$

Assume continue for k times

$T(n) = T(n-k) + k.c$

$n=k$

$T(n) = T(0) + n = n+1$

$T(n) = \Theta(n)$

b) Algorithm alg2(X[l...r])

    if (l==r) return X[l] $\rightarrow 1$

    else

    flr = floor((l+r)/2) $\rightarrow 1$

    tmp1 = alg2(X[l...flr]) $\rightarrow T(n/2)$

    tmp2 = alg2(X[flr+1... r]) $\rightarrow T(n/2)$

    if (tmp1 <= tmp2) return tmp1 $\rightarrow 1$

    else return tmp2 $\rightarrow 1$

$T(n) = 2T(\frac{n}{2}) + c$

Master Theorem $\rightarrow$ $a=2$, $b=2$, $d=0$ $\rightarrow T(n) = \Theta(n^{\log_b a}) = \Theta(n)$

Two algorithm has same time complexity therefore, both algorithms give results in the same time. It wouldn't matter which algorithm you should use.

Question - 2

Algorithm polynomial(P[0....n - 1], x)

```
    result = 0
    for i from n to 0 do
        power = 1
        for j from 0 to i do
            power = power * x
        result = result + P[i] * power
    return result
```

Time Complexity:

$$T(n) = \sum_{i=0}^{n} \sum_{j=1}^{i} 1 = \sum_{i=0}^{n} i = \frac{n(n+1)}{2} = n^2 \in \Theta(n^2)$$

$T(n) = \Theta(n^2)$

Question - 3

Algorithm count_str(str, length, first, last)

```
    count = 0
    for i from 0 to length do
        if str[i] == first do
            for j from i to length do
                if str[j] == last
                    count = count + 1
    return count
```

Time Complexity:

$$T(n) = \sum_{i=0}^{n} \sum_{j=i}^{n} 1 = \sum_{i=0}^{n} (n+1-i) = \Big(\underbrace{(n+1) + n + (n-1) + (n-2) \ldots}_{n \text{ times}}\Big)$$

$T(n) = \Theta(n^2)$

CamScanner ile tarandı

Question-4

```
Algorithm closestDistance(points, length)
    distance = infinty
    for i from 0 to length do
        for j from i+1 to length do
            distance = Distance(points[i], points[j])
    return distance
```

Given Function

Time complexity:

$$T(n) = \sum_{i=0}^{n} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n} (n-1) = \frac{n(n-1)}{2} \in \Theta(n^2)$$

$$T(n) = \Theta(n^2)$$

Question - 5

a) Algorithm max_clusters(branches[0....length-1], length)

```
max_profit = 0
clusters = [0, 0]
for i from 0 to length do
    profit = 0
    for J from i to length do
        profit = profit + branches[J]
        if profit >= max_profit do
            max_profit = profit
            clusters[0] = i
            clusters[1] = J
return clusters
```

Time Complexity:

$$\sum_{i=0}^{n} \sum_{J=i}^{n} 1 = \sum_{i=0}^{n} (n+1-i) \in \Theta(n^2)$$

$T(n) = \Theta(n^2)$

Question - 5

b)
```
Algorithm maxClusters (branches, low, middle, high)
    profit = 0
    left_sum = -999999
    for i from middle to low -1 do
        profit = profit + branches [i]
        if (left_sum < profit):
            left_sum = profit

    profit = 0
    right_sum = -999999
    for i from middle + 1 to high + 1 do
        profit = profit + branches [i]
        if right_sum < profit do
            right_sum < profit
    maximum = max (left_sum + right_sum, left_sum, right_sum)
    return maximum

Algorithm findProfit (branches, low high)
    if low == high                                       ) --> Θ(1)
        return branches [low]                            )
    middle = (low + high) // 2                                --> Θ(1)
    first = findProfit (branches, low, middle)                --> T(n/2)
    second = findProfit (branches, middle+1, high)            --> T(n/2)
    third = maxClusters (branches, low, middle, high)          --> Θ(n)
    maximum = max (first, second, third)                      --> Θ(1)
    return maximum
```

Time Complexity:

$T(n) = T(n/2) + T(n/2) + \Theta(n)$

$T(n) = 2T(n/2) + \Theta(n)$