PartA:

1. Since given that sequence a and b are concave, which means that:

1) a_i - a_{i-1} $\geq a_{i+1}$ - a_i
2) b_i - $b_{i-1} \geq$ b_{i+1} - b_i

In order to prove that z_i is also a concave function, we need to prove that:

3) z_i - z_{i-1} $\geq z_{i+1}$ - z_i $= 2z_i$ $\geq z_{i+1}$+z_{i-1}

By rearranging equation 1) and 2), we get:

4) $2a_i$ $\geq a_{i+1}$+a_{i-1}
5) $2b_i$ $\geq b_{i+1}$+b_{i-1}

Because z_i=a_i+$b_{\mathrm{k}-1}$, we plug in (k-i) to equation 2), we will get:

6) $2b_{k-i}$ $\geq b_{k-i+1}$+b_{k-i-1}

If we add equation 4) and 6) together, we will get:

7) $2a_i$ + $2b_{k-i}$ $\geq a_{i+1}$+a_{i-1} + b_{k-i+1}+b_{k-i-1}

= $2z_i \geq z_{i+1}$+z_{i-1}

2. Because c_k = a_i + $b_{\mathrm{k}-i}$

➔ O(logK)

We have an array of a -> a[n], and an array of b -> b[n], and we need to find the largest c_k possible. And because both a and b are concave, their sum should look like a somewhat concave down parabola, and the maxima is likely to be found near the vertex, if the concave function is skewed, we can recurve into either left or right side of the vertex, and recursively look for the vertex.

$c_{k\ max}$ = $a[n]_{max}$ + $b[n]_{max}$

And assume that

Algo design:

VertexFinder (left, right, size, c_k)
    // if there's only a single element
    If (left == right)
        Return c_k[left]
    // record the middle point
    mid = (left+right)/2;
    // access the middle value
    sum = $c_k[mid]$
    // if the sum at middle point is larger than its neighbor, the we
    // have successfully identified c_{max}
    if (sum >=$c_k[mid-1]$ && sum >=$c_k[mid+1]$)
        return sum;
    // if the left neighbor is larger, then we recurse into the left half
    if (sum < $c_k[mid-1]$)
        VertexFinder(left, mid-1, size, c_k)
    // if the right neighbor is larger, then we recurse into the left half
    Else if (sum < $c_k[mid+1]$)

VertexFinder(mid, right, size, c_k)

Complexity Analysis:

T(n) = T(n/2) + const -> O(c*log_2n) -> O(log_2K)

Biref Proof of Correctness:
As shown in question (a) that the c_k is a concave function, and it is supposed to be a somewhat concave down quadratic looking function whether or not it is skewed. We can find the max by looking for its vertex, in which we can use D&C algo. This will always work for any c of size k due to its already proved property of concavity.

PartB:

4. If b is concave, then we have b_i - $b_{i-1} \geq$ b_{i+1} - b_i and 2b_i $\geq b_{i+1}$+b_{i-1}

in another word, $b_{i-1} <$ $b_i <$ b_{i+1} for a particular i

We can use proof by contradiction to prove this.

Given that i(k) <= i(k+1), we first start off assuming i(k) >= i(k+1)

Let max = i(k), max+1 = i(k+1) and we will have the equation

1) c_{max} = a_{max} + b_{k-max}
2) c_{max+1} = a_{max+1} + $b_{K+1-max-1}$

Because we assume that i(k) > i(k+1), we have:

3) a_{max} + b_{k-max} > a_{max+1} + b_{K-max}

However, we are given that a is a non-decreasing sequence, therefore, a_{max+1} should be larger than a_{max}, so we should have:

3) 3) a_{max} + b_{k-max} < a_{max+1} + b_{K-max}

By 3) and 4), we have proven the validity of i(k) as a non-decreasing sequence by contradiction

5.