Q 1.1)

Since the view count integer array is sorted in descending order. Which means at index $p-1$, there is p posts (because index starts at 0) that have at least view count of that $p-1^{th}$ post.

In order to determine whether that user's popularity is at least p, we could compare p with the value in the array at index $p-1$. If the element at index $p-1$ is at least p then the user's popularity is also at least p.

By using hash table storing each user's information having each user's id as a kay, it would take constant time to search for a specific user. Then there is an array of user's popularity in each user's information. By simply using the algorithm mentioned above, it would also take constant time to search to a specific index of an array.

Overall time complexity takes $O(1)+O(1)=O(1)$ since searching in a hash table using a key and searching in an array by a specific index both take constant time.

Q 1.2)

Since we know how to determine whether that user's popularity is at least p using previous question's algorithm.

By using binary search but with a little different algorithm. Using any user's view count integer array that is already been sorted in descending order. Set the low index to the first element of the array and the high index to the last element then set the middle index to the average of the low and high indices just like normal binary search.

However, if $index+1$ is less than or equal to the element at the middle index, set the low index to the middle index (since this might not be the biggest p so we would search for a possible bigger p).

If $index+1$ is greater than the element at the middle index (which means that there are less than $index+1$ posts with at least $(index+1)k$ views so we need to search for a smaller p), set the high index to the middle index - 1.

Eventually after repeating each step until the low index is next to the high index or both happens to be the same index. If $high\ index+1$ is greater than or equal to the element at the high index then $high\ index+1$ would be the user's popularity index p, p is $low\ index+1$ otherwise.

Since all elements with smaller index would have a higher view count because the array is sorted in descending order. Which means that now we would have the most p posts with at least $(p)k$ views.

This is done in $O(\log n)$ even though this algorithm is different from binary search but the way it is searching and marking new low and high indexes are almost the same. So that the time complexity is the same as binary search in worst case (since this wouldn't stop until the condition is satisfied unlike binary search that could stop at the first search when the element at the index is the target element). We are halving number of search elements until we get the last element, so it is $O(\log n)$.