BRUTE:

def missingK(vec: List[int], n: int, k: int) -> int:

    maxi=vec[n-1]

    for i in range(maxi+1):

        if (i+1) not in vec:

            k=k-1

            if k==0:

                return (i+1)

    j=maxi+1

    while(k>0):

        j+=1

        k=k-1

    return j

OPTIMAL:

from typing import \*

def missingK(vec: List[int], n: int, k: int) -> int:

    s=0

    e=n-1

    for i in range(n):

        if vec[i]>=0:

            s=i

            break

    while(s<=e):

        mid=(s+e)//2

        mis=vec[mid]-(mid+1)#as in it is a sorted array and

        #the elemnt that was supposed to be there vs that that is there

        #Ex in [2,4,5,7] PosOf4=1 therefore 4-1-1=2 missing values(1,3)

        #since number of missing values are less i can eliminate left part

        if mis<k:

            s=mid+1#s=2

        else:

            e=mid-1

    #Again at 7 s==e==3 missing values 7-3-1=3(1,3,6) and e=2 s=3 breaks out

    #more\_missing\_numbers = k - (vec[high] - (high+1)).

    #Now, we will simply add more\_missing\_numbers to the preceding neighbor i.e. vec[high] to get the kth missing number.

    #kth missing number = vec[high] + k - (vec[high] - (high+1))

        #=  vec[high] + k - vec[high] + high + 1

        #= k + high + 1.

    return e+k+1#would take take if elemnt was outside array that was missing