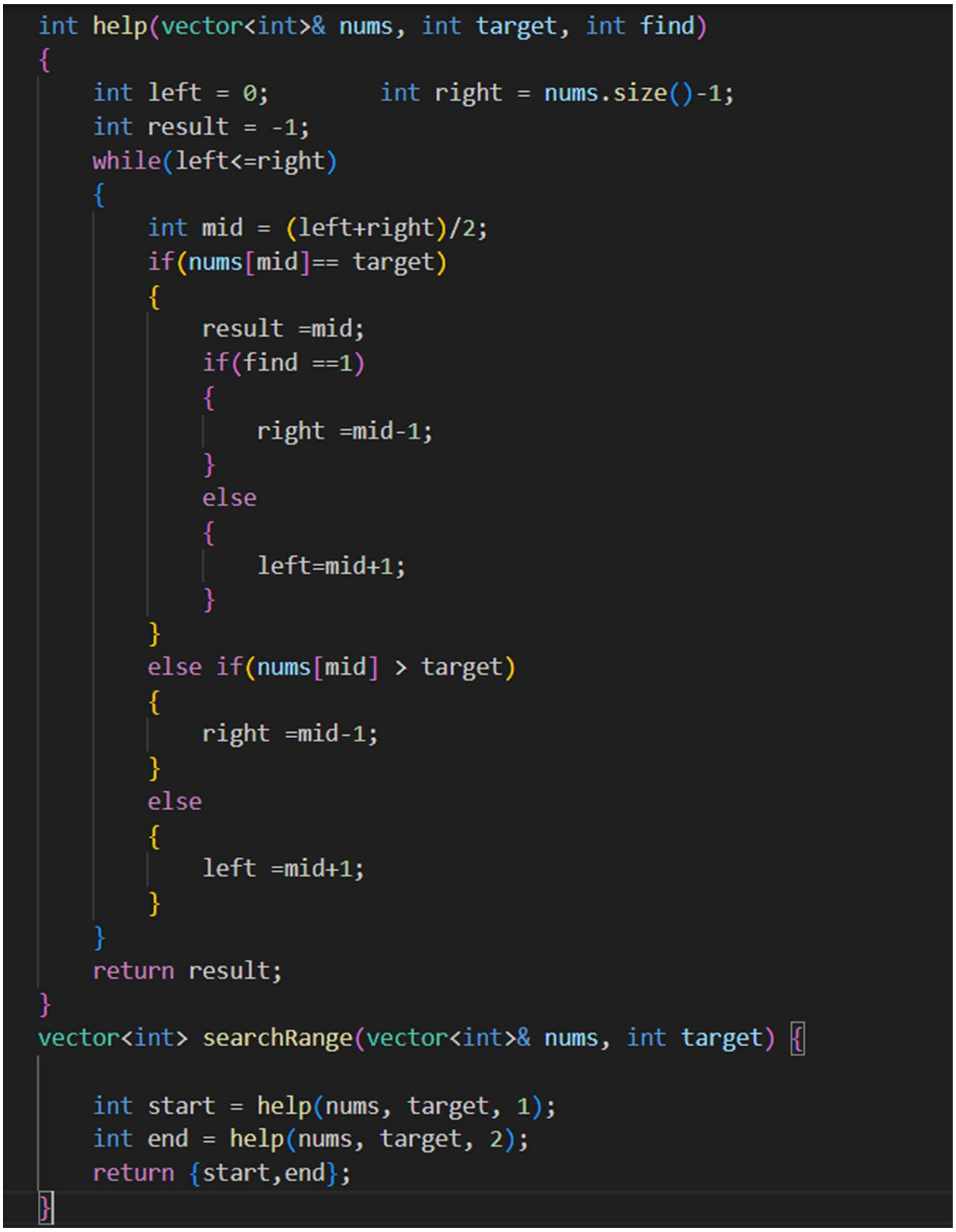
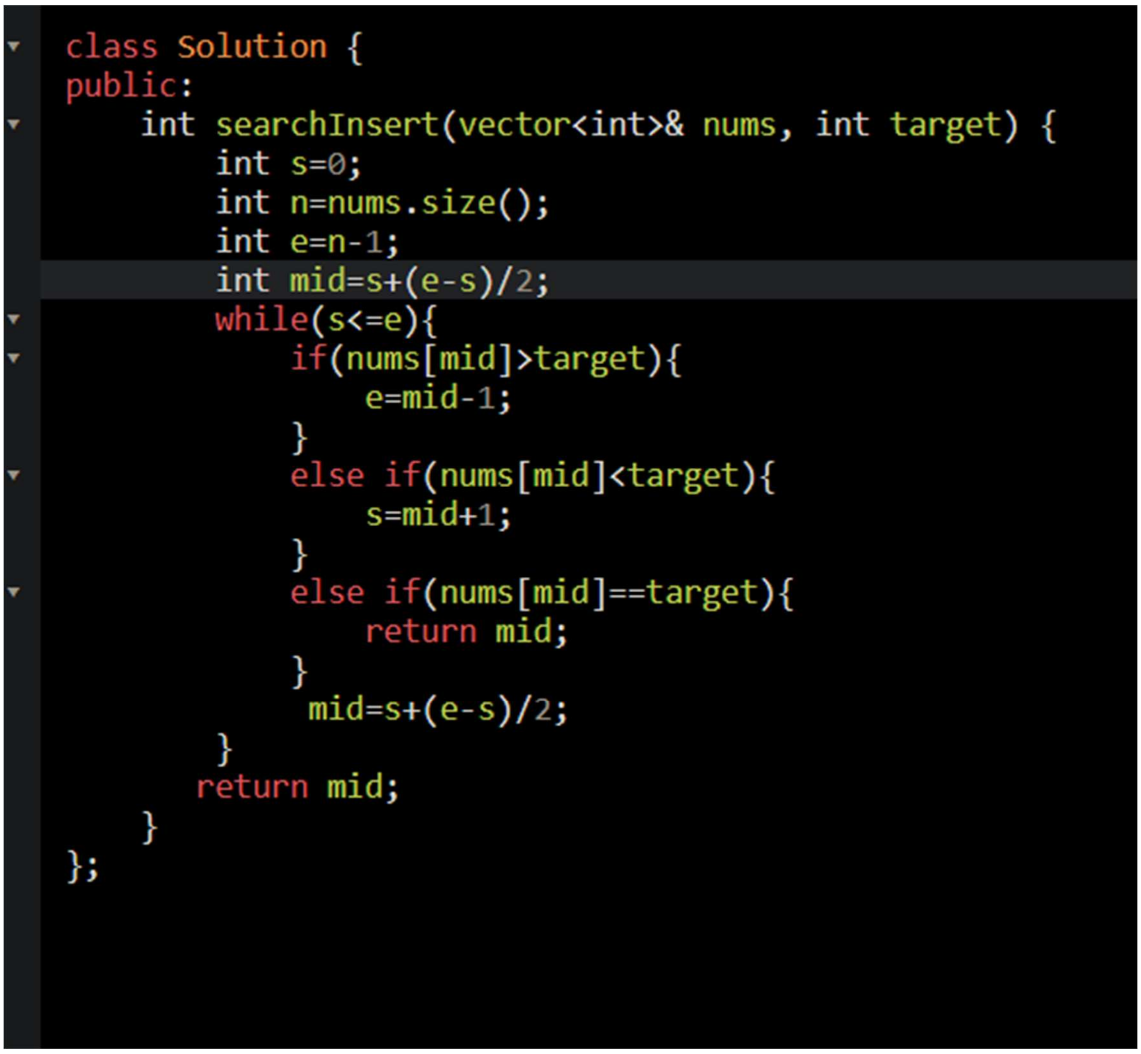
DAY 31/180

Q1- Find first and last position of element in a sorted array.



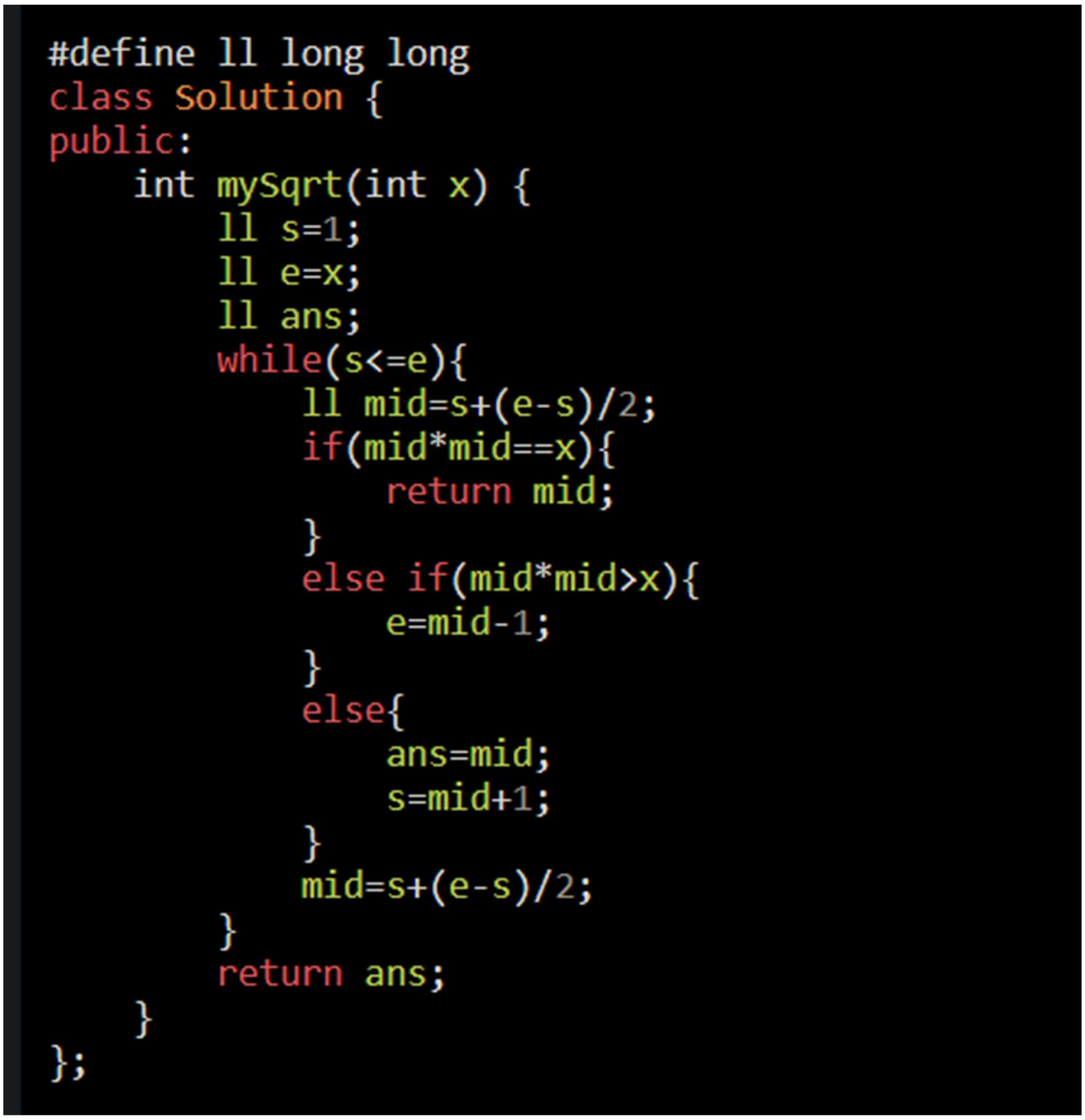
```cpp
int help(vector<int>& nums, int target, int find)
{
    int left = 0;         int right = nums.size()-1;
    int result = -1;
    while(left<=right)
    {
        int mid = (left+right)/2;
        if(nums[mid]== target)
        {
            result =mid;
            if(find ==1)
            {
                right =mid-1;
            }
            else
            {
                left=mid+1;
            }
        }
        else if(nums[mid] > target)
        {
            right =mid-1;
        }
        else
        {
            left =mid+1;
        }
    }
    return result;
}
vector<int> searchRange(vector<int>& nums, int target) {

    int start = help(nums, target, 1);
    int end = help(nums, target, 2);
    return {start,end};
}
```

Q2-Search Insert Position



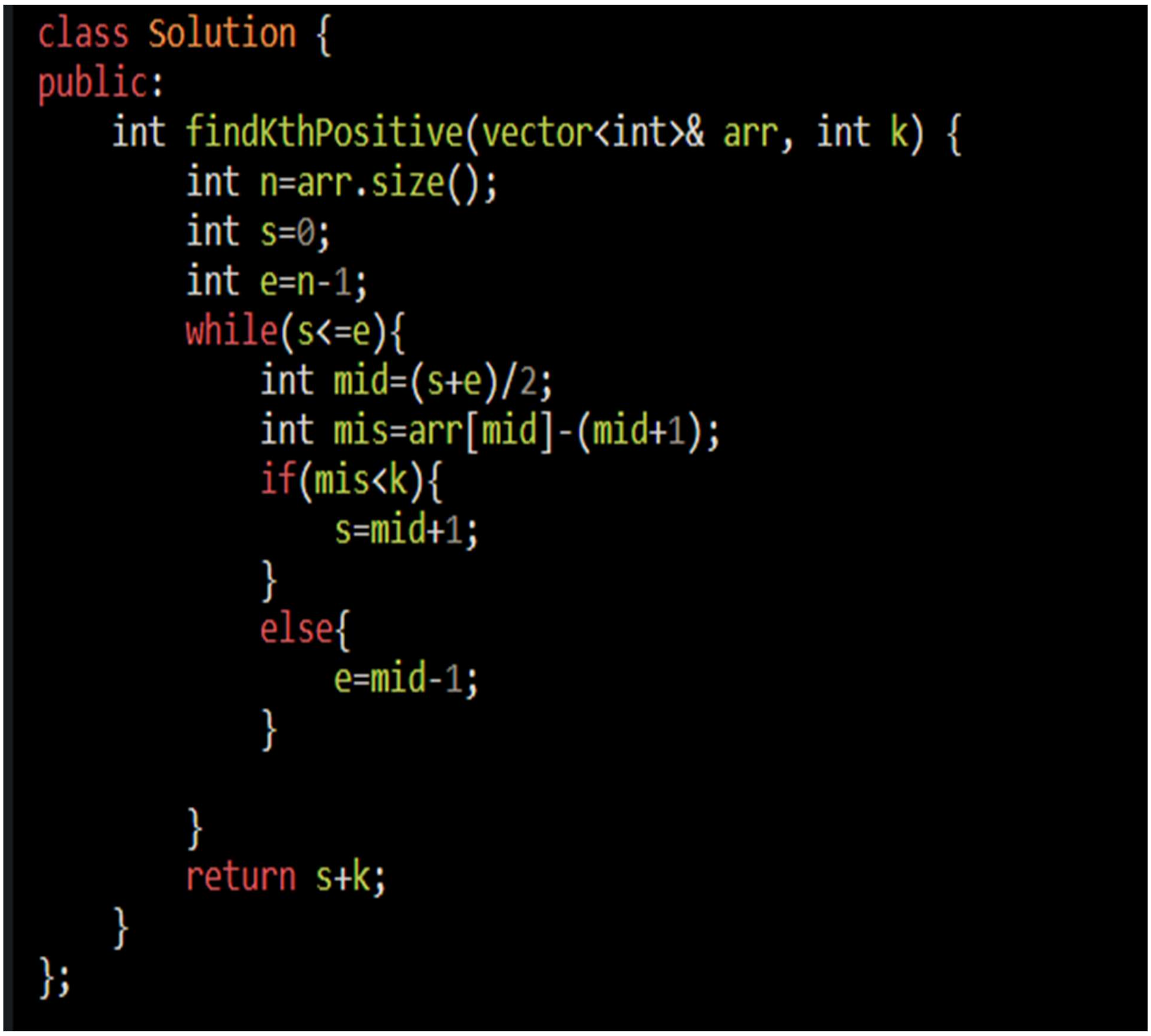
```cpp
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int s=0;
        int n=nums.size();
        int e=n-1;
        int mid=s+(e-s)/2;
        while(s<=e){
            if(nums[mid]>target){
                e=mid-1;
            }
            else if(nums[mid]<target){
                s=mid+1;
            }
            else if(nums[mid]==target){
                return mid;
            }
             mid=s+(e-s)/2;
        }
       return mid;
    }
};
```

Q3- Sqrt(X) by binary search.



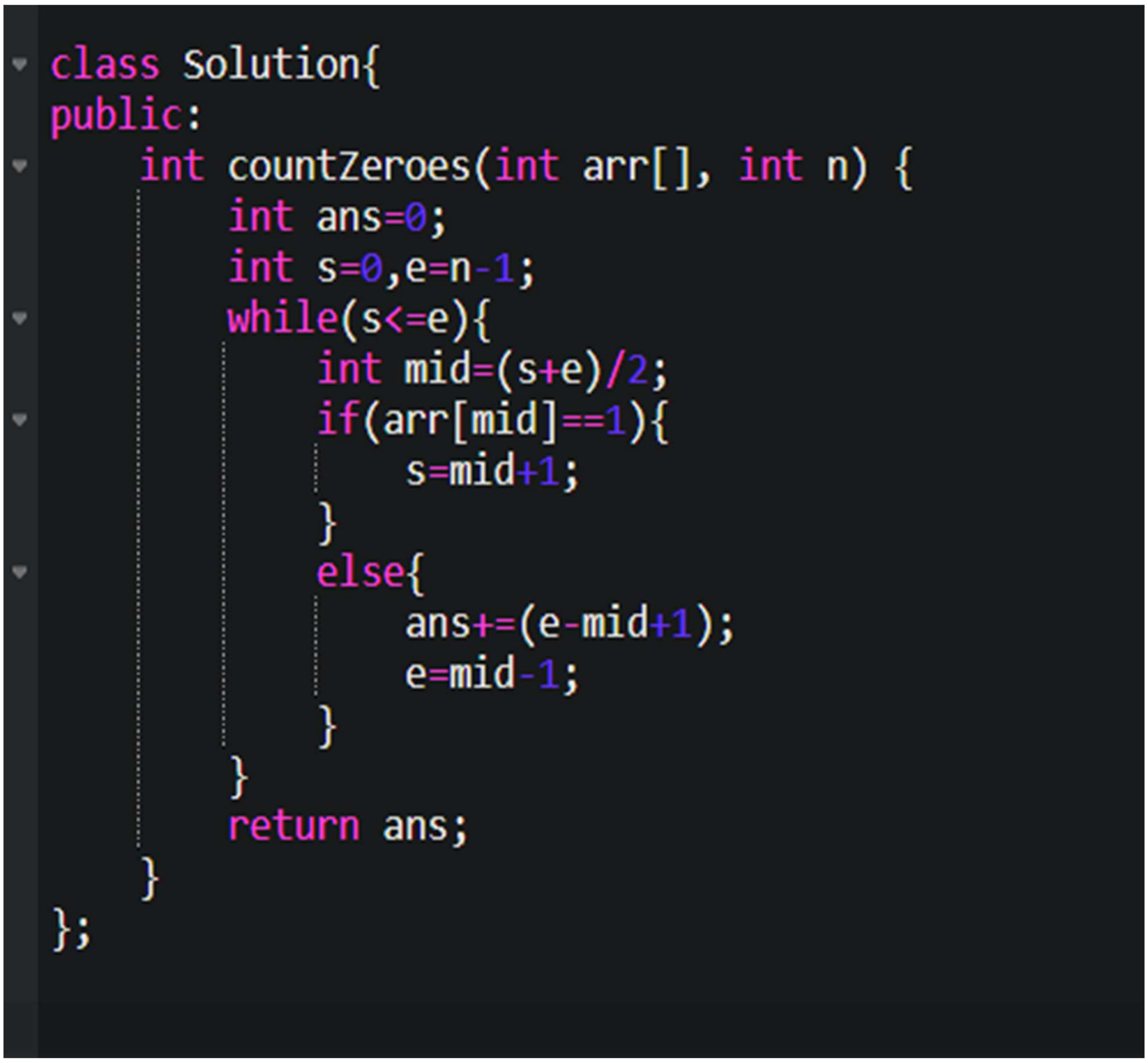
```cpp
#define ll long long
class Solution {
public:
    int mySqrt(int x) {
        ll s=1;
        ll e=x;
        ll ans;
        while(s<=e){
            ll mid=s+(e-s)/2;
            if(mid*mid==x){
                return mid;
            }
            else if(mid*mid>x){
                e=mid-1;
            }
            else{
                ans=mid;
                s=mid+1;
            }
            mid=s+(e-s)/2;
        }
        return ans;
    }
};
```

Q4- Kth missing positive number.



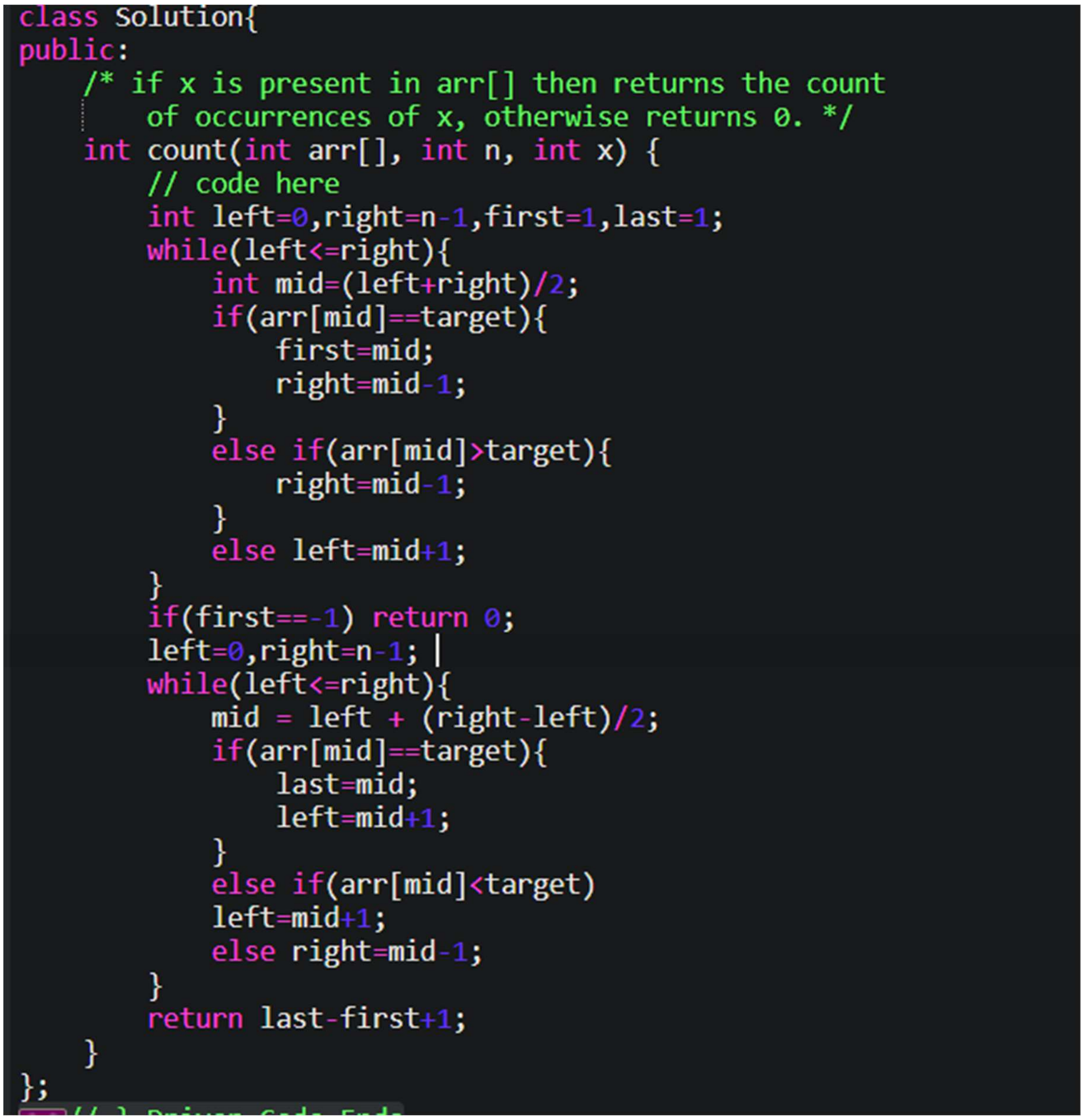
```
class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        int n=arr.size();
        int s=0;
        int e=n-1;
        while(s<=e){
            int mid=(s+e)/2;
            int mis=arr[mid]-(mid+1);
            if(mis<k){
                s=mid+1;
            }
            else{
                e=mid-1;
            }

        }
        return s+k;
    }
};
```

Q5- Count the Zeros



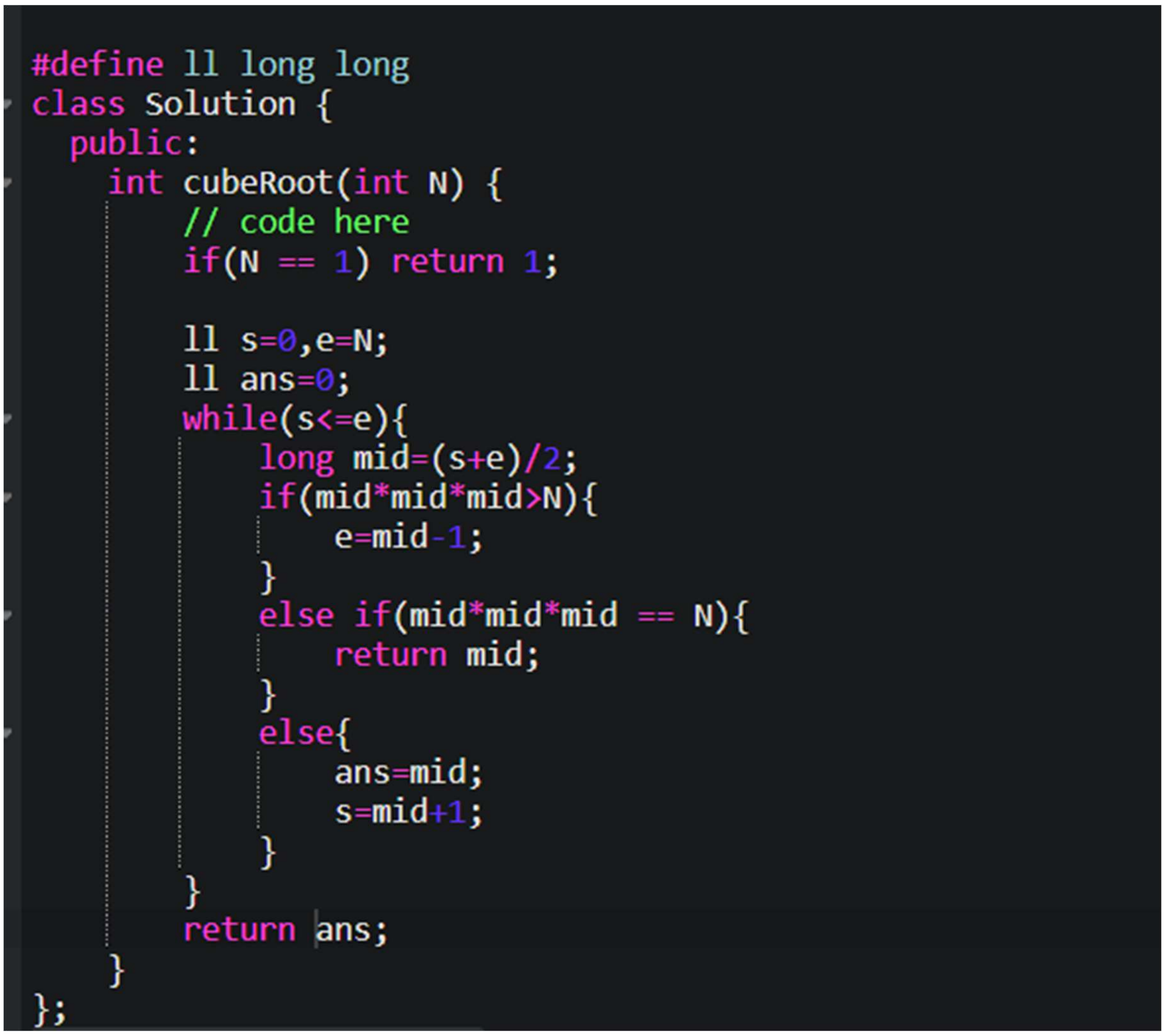
```
class Solution{
public:
    int countZeroes(int arr[], int n) {
        int ans=0;
        int s=0,e=n-1;
        while(s<=e){
            int mid=(s+e)/2;
            if(arr[mid]==1){
                s=mid+1;
            }
            else{
                ans+=(e-mid+1);
                e=mid-1;
            }
        }
        return ans;
    }
};
```

Q6- Number of Occurrences.



```
class Solution{
public:
    /* if x is present in arr[] then returns the count
        of occurrences of x, otherwise returns 0. */
    int count(int arr[], int n, int x) {
        // code here
        int left=0,right=n-1,first=1,last=1;
        while(left<=right){
            int mid=(left+right)/2;
            if(arr[mid]==target){
                first=mid;
                right=mid-1;
            }
            else if(arr[mid]>target){
                right=mid-1;
            }
            else left=mid+1;
        }
        if(first==-1) return 0;
        left=0,right=n-1;
        while(left<=right){
            mid = left + (right-left)/2;
            if(arr[mid]==target){
                last=mid;
                left=mid+1;
            }
            else if(arr[mid]<target)
            left=mid+1;
            else right=mid-1;
        }
        return last-first+1;
    }
};
// } Driver Code Ends
```

Q7- Cube Root of a number.



```
#define ll long long
class Solution {
  public:
    int cubeRoot(int N) {
        // code here
        if(N == 1) return 1;

        ll s=0,e=N;
        ll ans=0;
        while(s<=e){
            long mid=(s+e)/2;
            if(mid*mid*mid>N){
                e=mid-1;
            }
            else if(mid*mid*mid == N){
                return mid;
            }
            else{
                ans=mid;
                s=mid+1;
            }
        }
        return ans;
    }
};
```