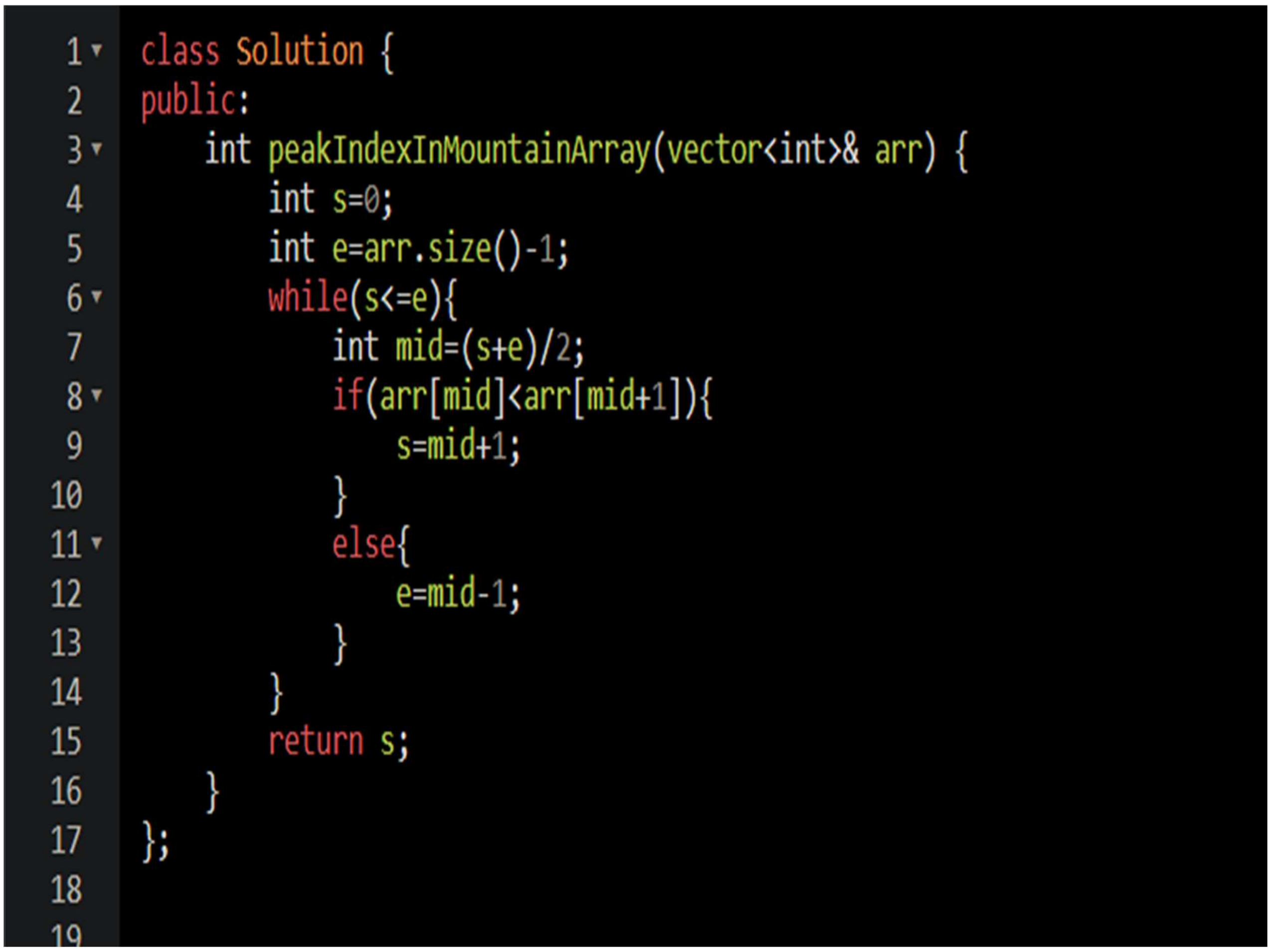
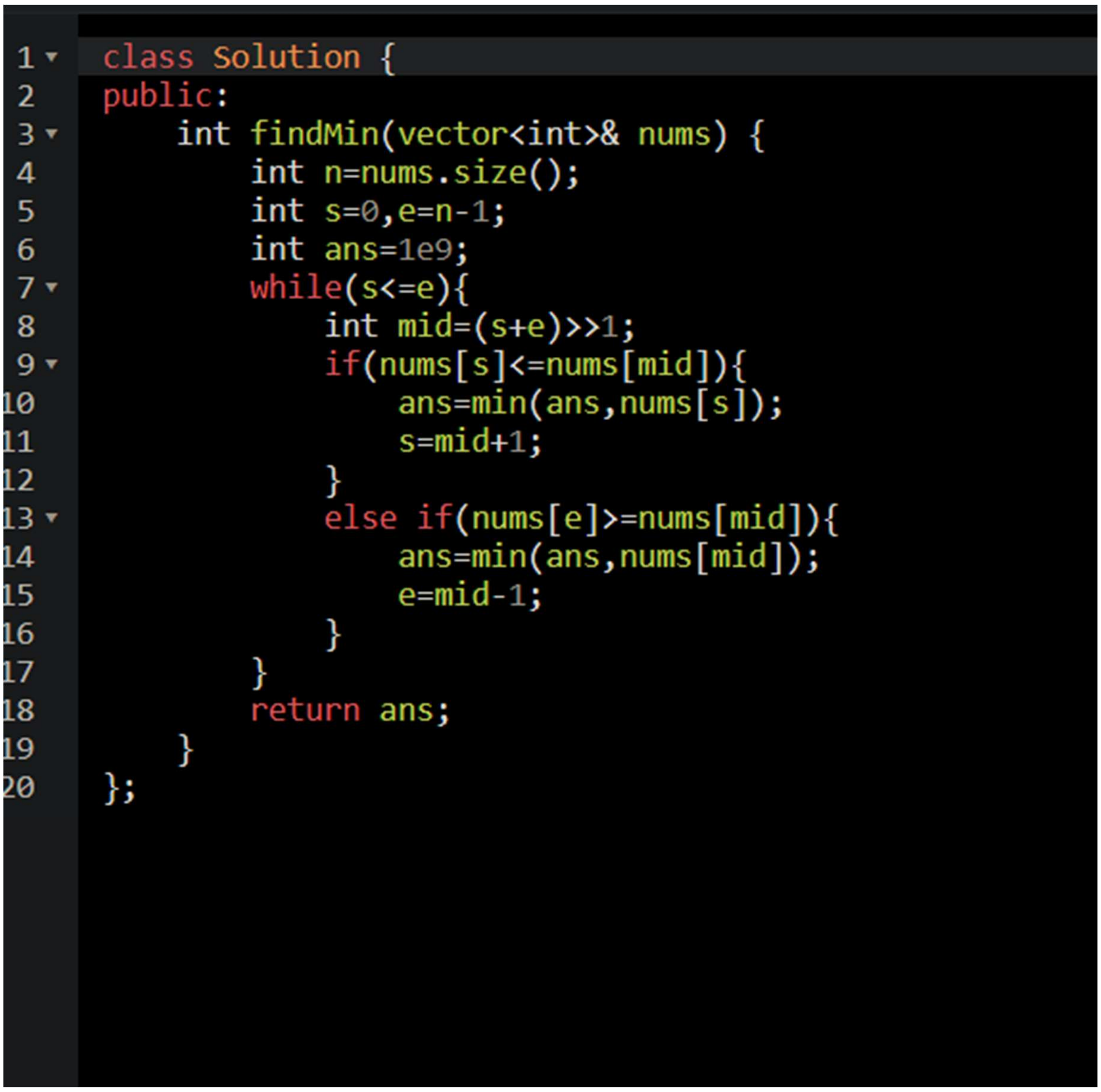
DAY 32/180

Q1- Peak Index in a Mountain Array



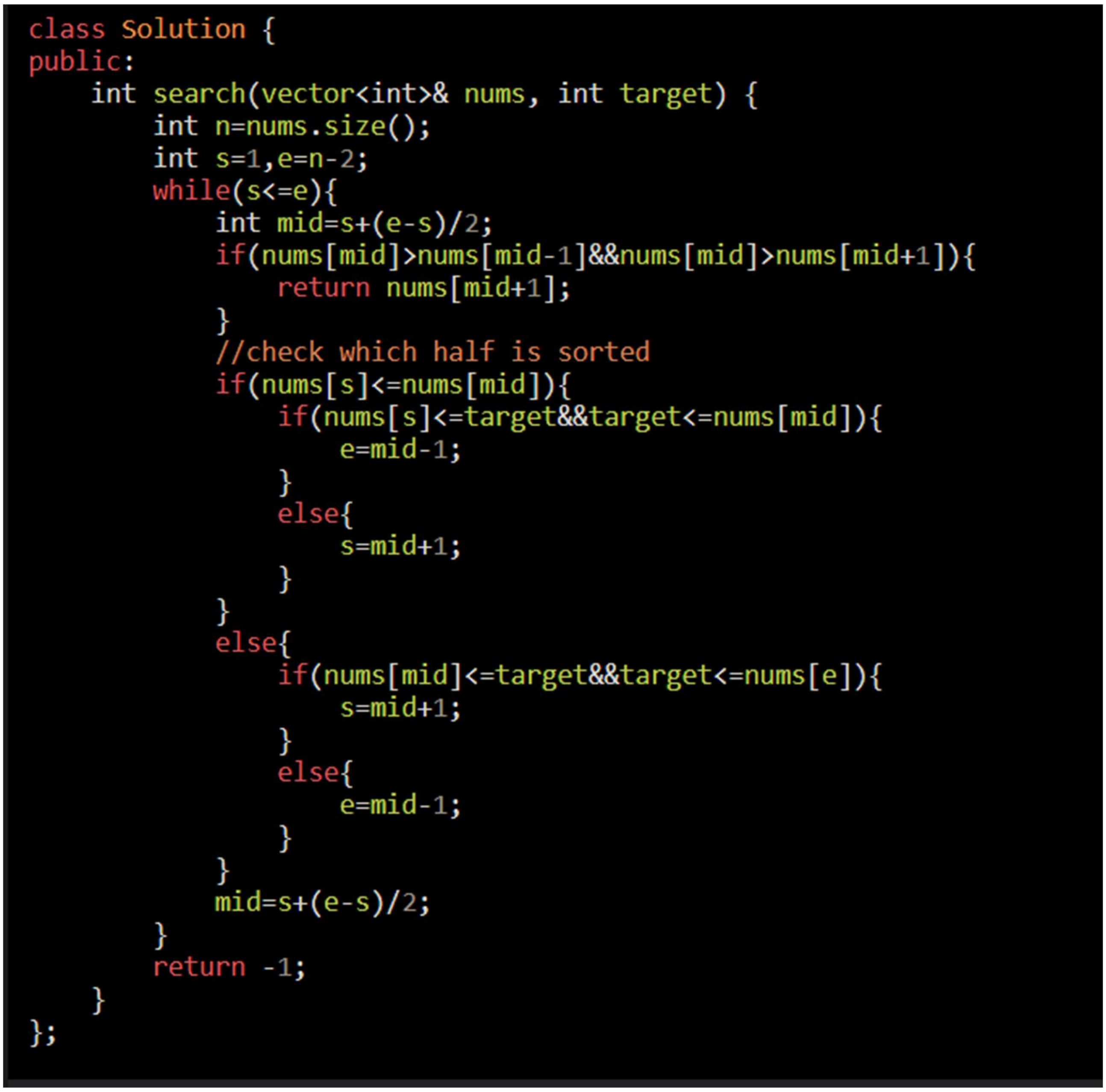
```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& arr) {
        int s=0;
        int e=arr.size()-1;
        while(s<=e){
            int mid=(s+e)/2;
            if(arr[mid]<arr[mid+1]){
                s=mid+1;
            }
            else{
                e=mid-1;
    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

Q2- Find Minimum in Rotated Sorted Array.



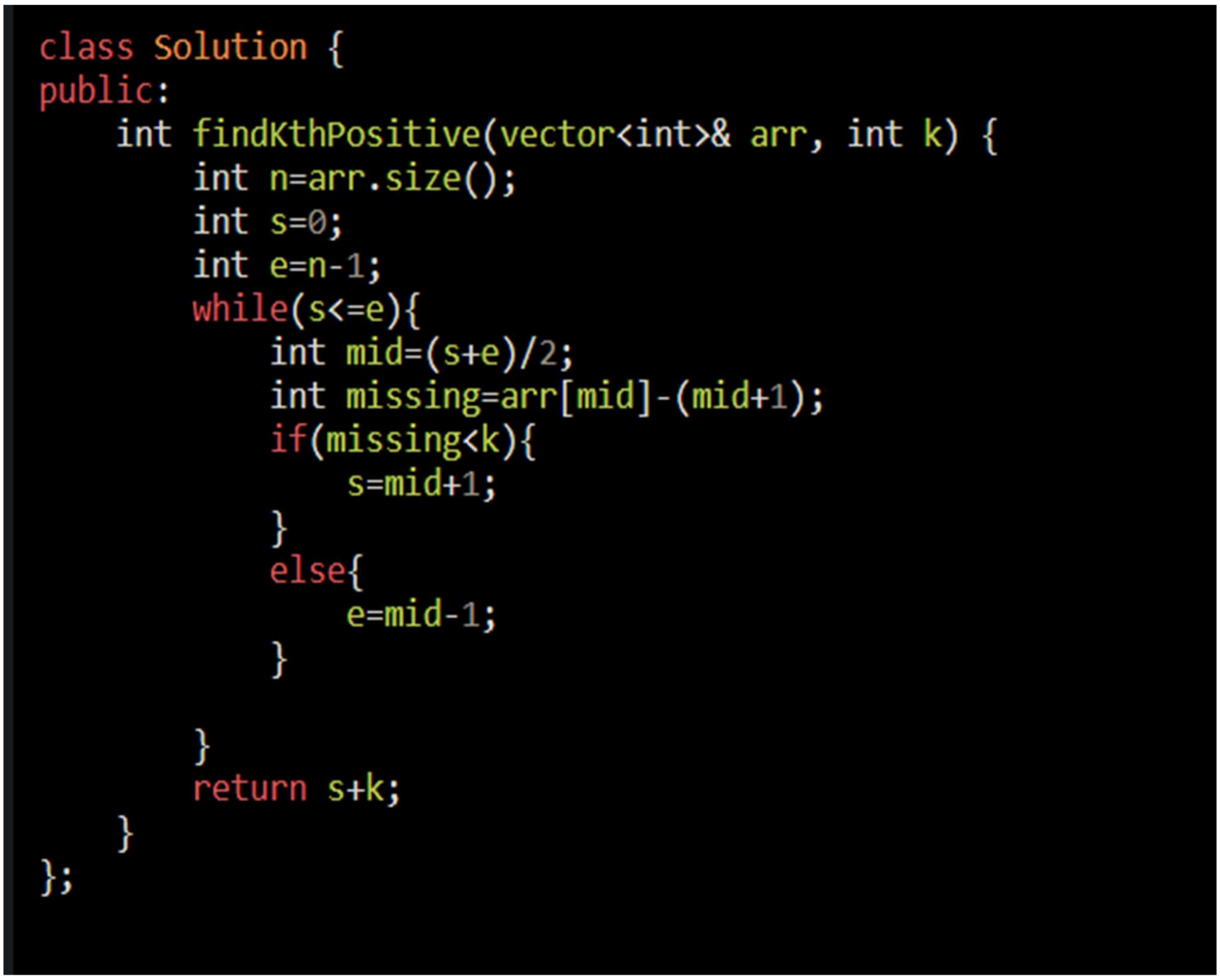
```
class Solution {
public:
    int findMin(vector<int>& nums) {
        int n=nums.size();
        int s=0,e=n-1;
        int ans=1e9;
        while(s<=e){
            int mid=(s+e)>>1;
            if(nums[s]<=nums[mid]){
                ans=min(ans,nums[s]);
                s=mid+1;
            }
            else if(nums[e]>=nums[mid]){
                ans=min(ans,nums[mid]);
                e=mid-1;
            }
        }
        return ans;
    }
};
```

Q3- Search in a Rotated Sorted Array.



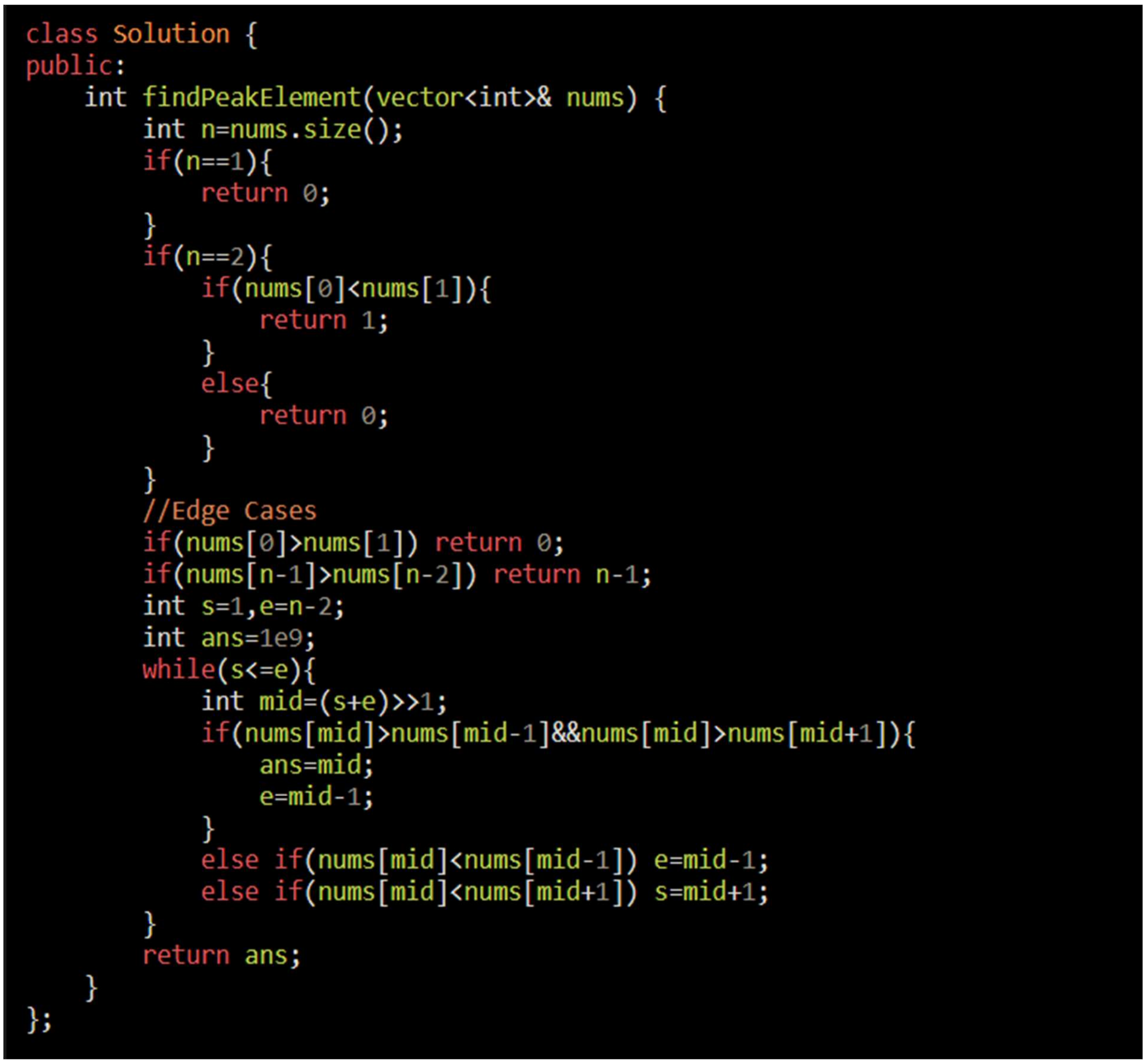
```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int n=nums.size();
        int s=1,e=n-2;
        while(s<=e){
            int mid=s+(e-s)/2;
            if(nums[mid]>nums[mid-1]&&nums[mid]>nums[mid+1]){
                return nums[mid+1];
            }
         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

Q4- Kth Missing Positive Number.



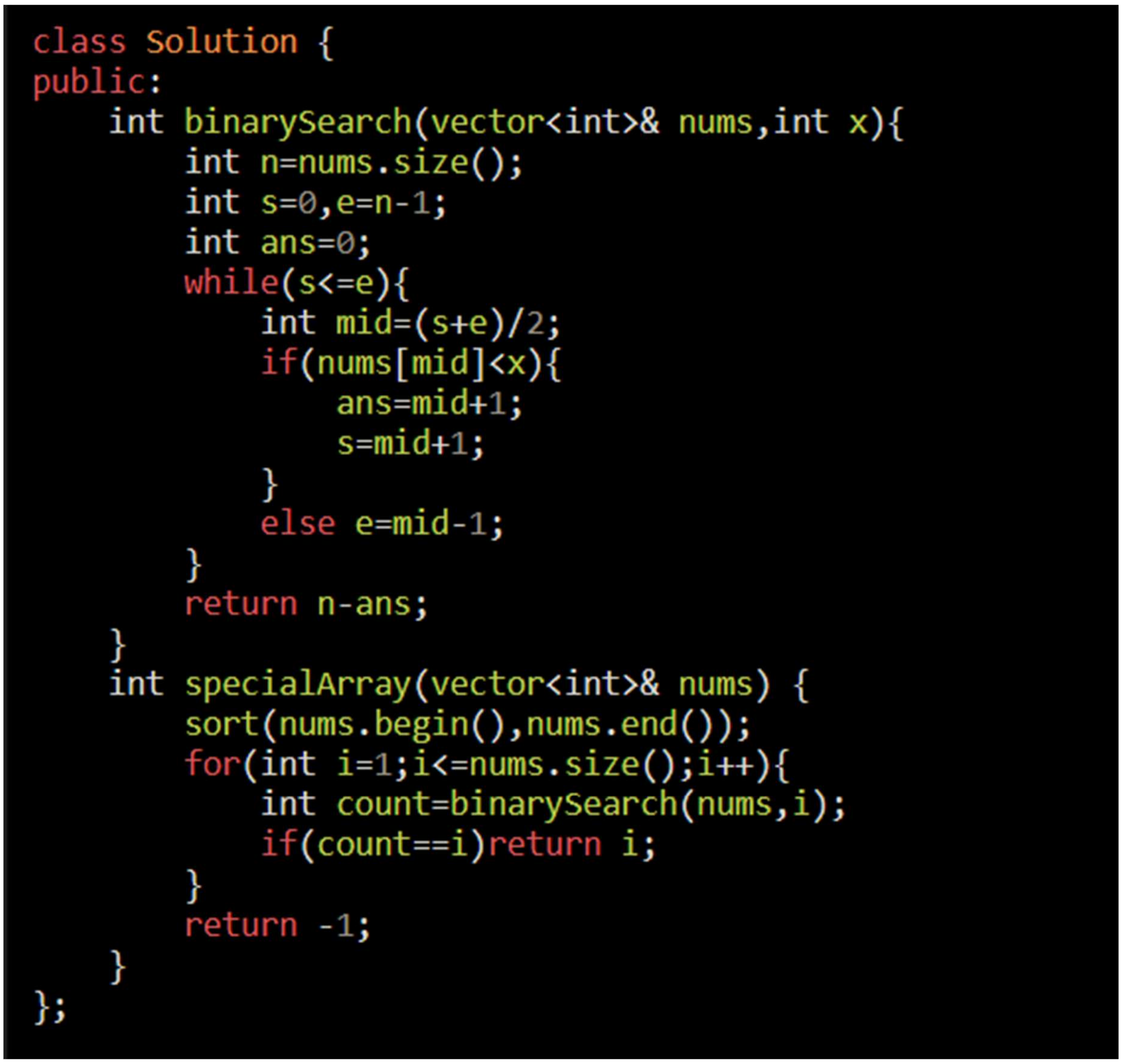
```
class Solution {
public:
    int findKthPositive(vector<int>& arr, int k) {
        int n=arr.size();
        int s=0;
        int e=n-1;
        while(s<=e){
            int mid=(s+e)/2;
            int missing=arr[mid]-(mid+1);
            if(missing<k){
                s=mid+1;
            }
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

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

Q5- Find Peak Element



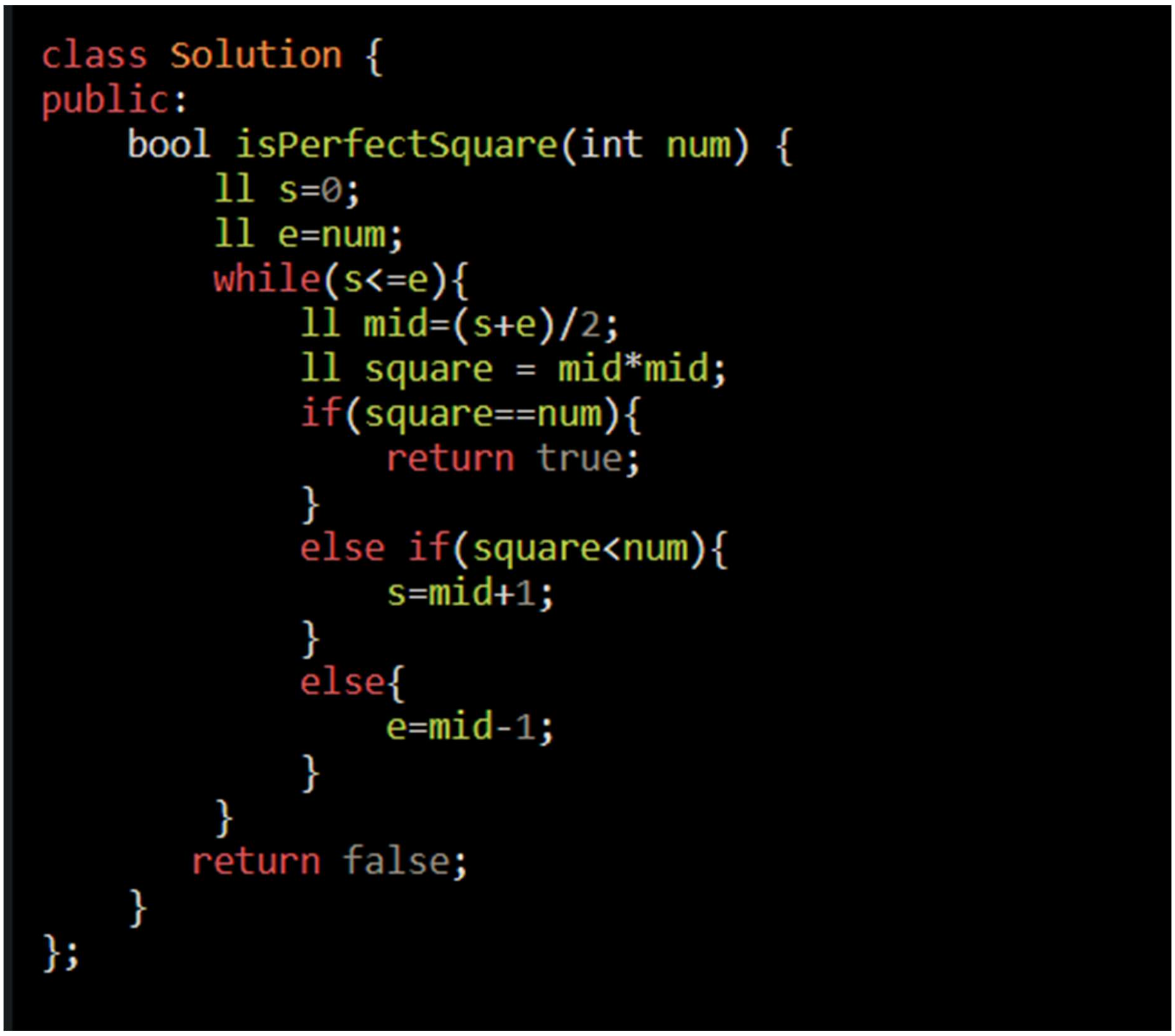
```cpp
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int n=nums.size();
        if(n==1){
            return 0;
        }
        if(n==2){
            if(nums[0]<nums[1]){
                return 1;
            }
            else{
                return 0;
            }
        }
        //Edge Cases
        if(nums[0]>nums[1]) return 0;
        if(nums[n-1]>nums[n-2]) return n-1;
        int s=1,e=n-2;
        int ans=1e9;
        while(s<=e){
            int mid=(s+e)>>1;
            if(nums[mid]>nums[mid-1]&&nums[mid]>nums[mid+1]){
                ans=mid;
                e=mid-1;
            }
            else if(nums[mid]<nums[mid-1]) e=mid-1;
            else if(nums[mid]<nums[mid+1]) s=mid+1;
        }
        return ans;
    }
};
```

Q6- Special Array with X elements greater than X.



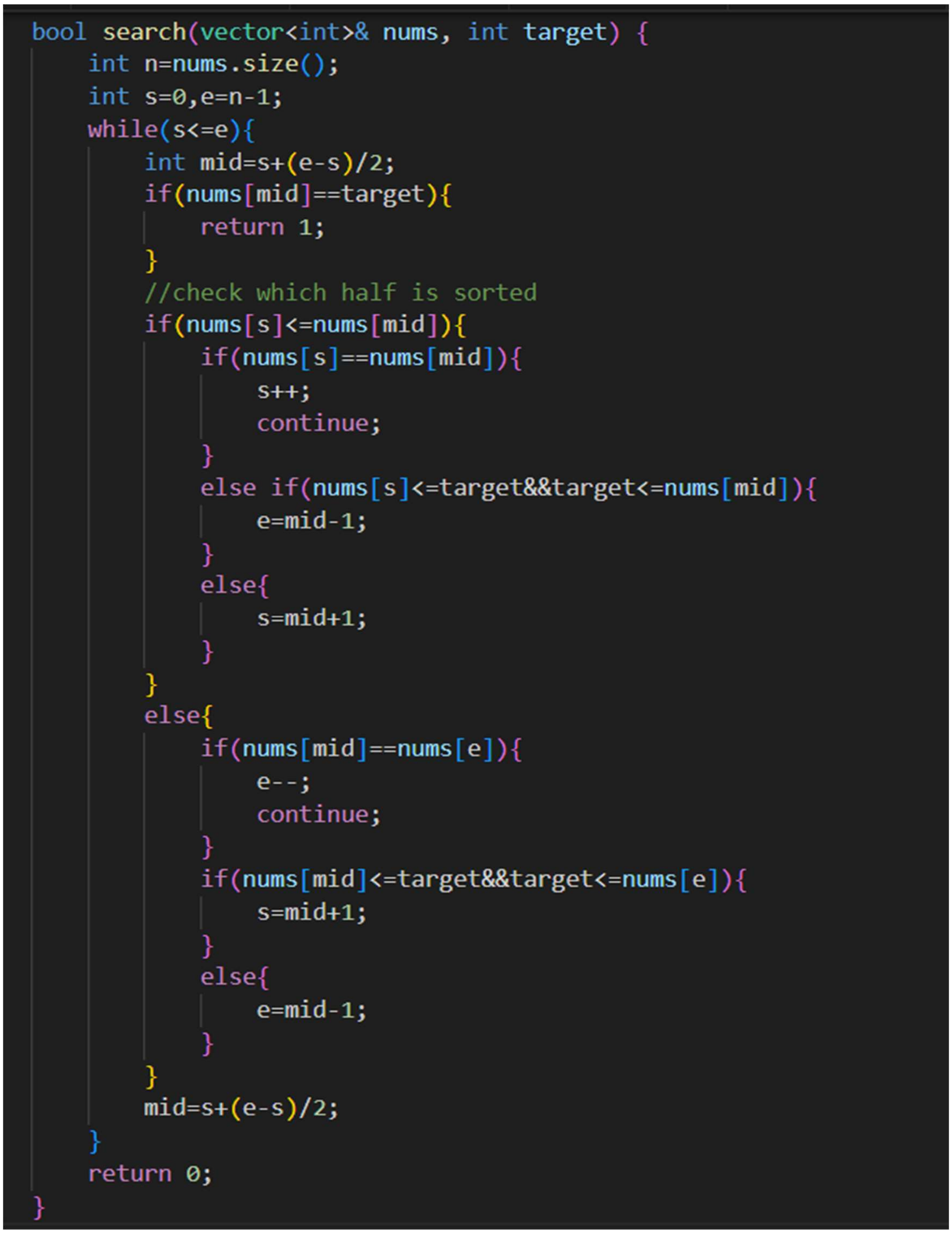
```cpp
class Solution {
public:
    int binarySearch(vector<int>& nums,int x){
        int n=nums.size();
        int s=0,e=n-1;
        int ans=0;
        while(s<=e){
            int mid=(s+e)/2;
            if(nums[mid]<x){
                ans=mid+1;
                s=mid+1;
            }
            else e=mid-1;
        }
        return n-ans;
    }
    int specialArray(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        for(int i=1;i<=nums.size();i++){
            int count=binarySearch(nums,i);
            if(count==i)return i;
        }
        return -1;
    }
};
```

Q7- Valid Perfect Square



```
class Solution {
public:
    bool isPerfectSquare(int num) {
        ll s=0;
        ll e=num;
        while(s<=e){
            ll mid=(s+e)/2;
            ll square = mid*mid;
            if(square==num){
                return true;
            }
            else if(square<num){
                s=mid+1;
            }
            else{
                e=mid-1;
            }
        }
       return false;
    }
};
```

Q8- Seach In Rotated Sorted Array – II



```cpp
bool search(vector<int>& nums, int target) {
    int n=nums.size();
    int s=0,e=n-1;
    while(s<=e){
        int mid=s+(e-s)/2;
        if(nums[mid]==target){
            return 1;
        }
        //check which half is sorted
        if(nums[s]<=nums[mid]){
            if(nums[s]==nums[mid]){
                s++;
                continue;
            }
            else if(nums[s]<=target&&target<=nums[mid]){
                e=mid-1;
            }
            else{
                s=mid+1;
            }
        }
        else{
            if(nums[mid]==nums[e]){
                e--;
                continue;
            }
            if(nums[mid]<=target&&target<=nums[e]){
                s=mid+1;
            }
            else{
                e=mid-1;
            }
        }
        mid=s+(e-s)/2;
    }
    return 0;
}
```