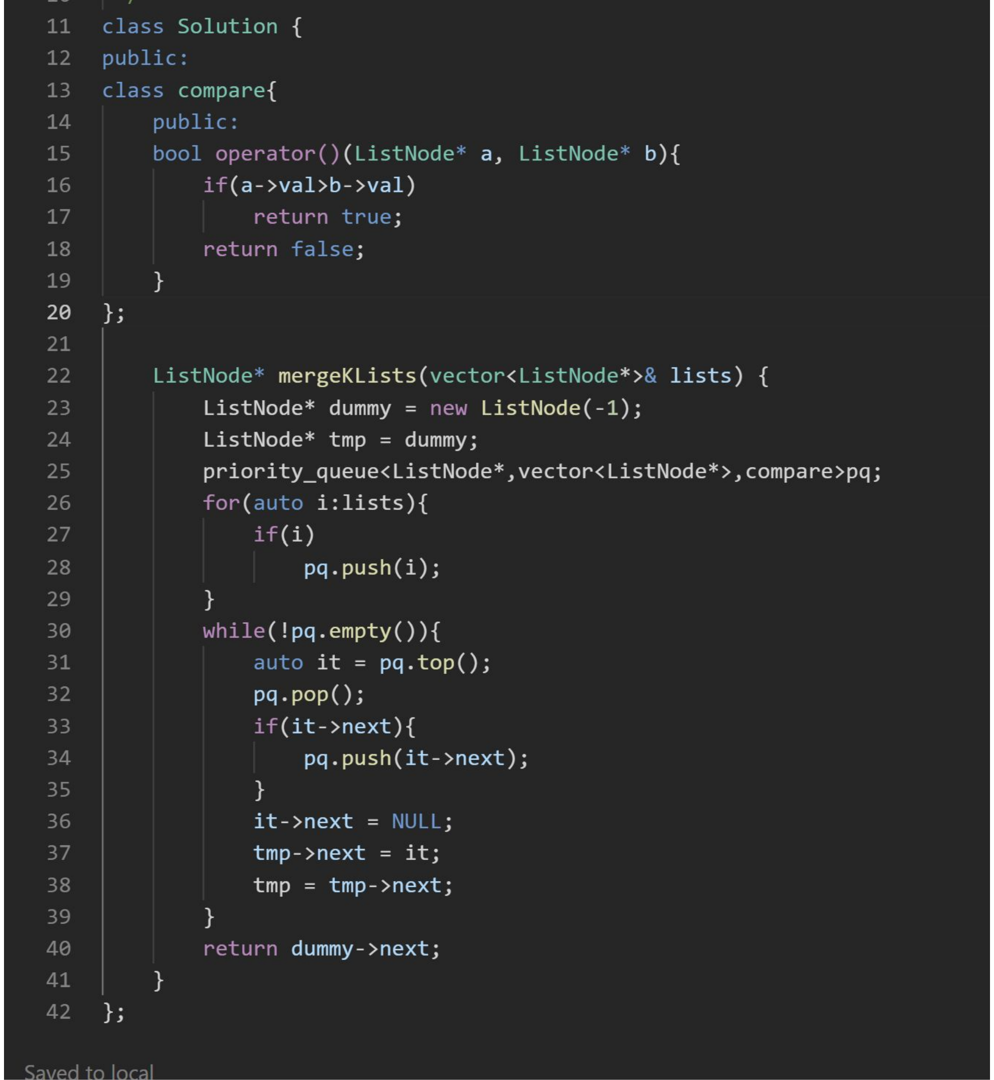
Heap Problems

Session 28

Question

Merge k sorted list

Link : https://leetcode.com/problems/merge-k-sorted-lists/

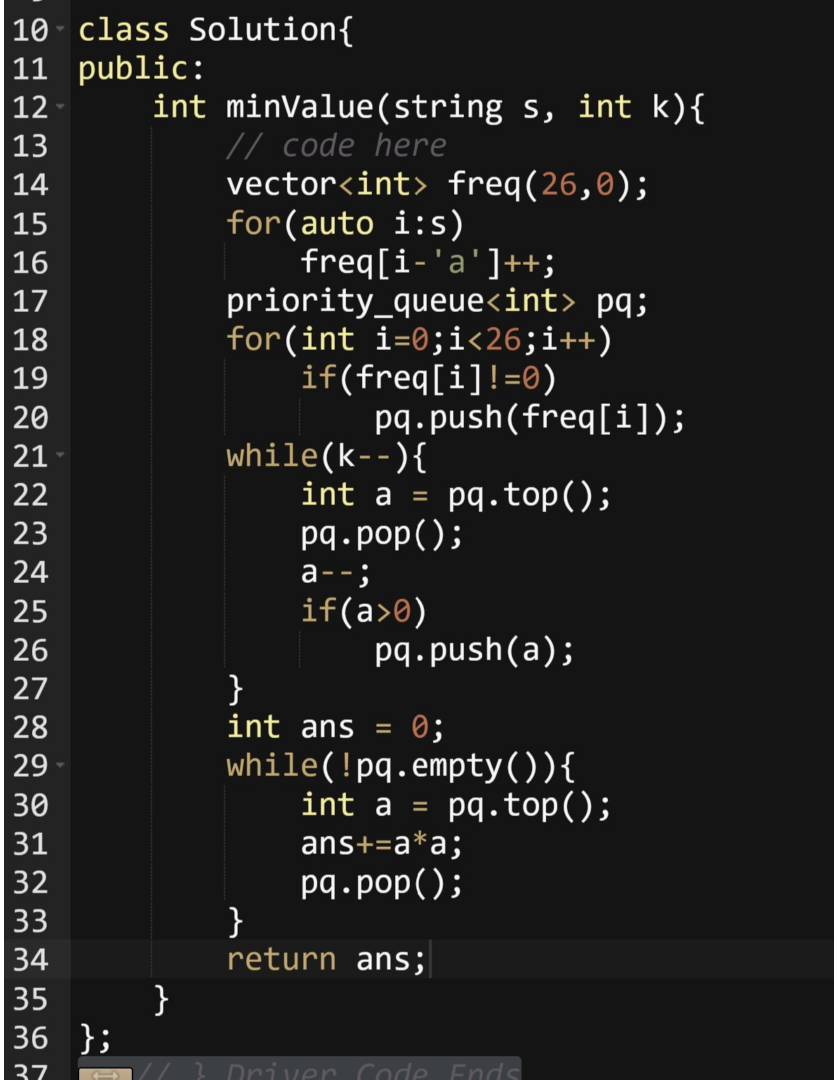


```
class Solution {
public:
class compare{
    public:
    bool operator()(ListNode* a, ListNode* b){
        if(a->val>b->val)
            return true;
        return false;
    }
};

    ListNode* mergeKLists(vector<ListNode*>& lists) {
        ListNode* dummy = new ListNode(-1);
        ListNode* tmp = dummy;
        priority_queue<ListNode*,vector<ListNode*>,compare>pq;
        for(auto i:lists){
            if(i)
                pq.push(i);
        }
        while(!pq.empty()){
            auto it = pq.top();
            pq.pop();
            if(it->next){
                pq.push(it->next);
            }
            it->next = NULL;
            tmp->next = it;
            tmp = tmp->next;
        }
        return dummy->next;
    }
};
```

Saved to local

Problems

- https://practice.geeksforgeeks.org/problems/game-with-string4100/1/?page=1&difficulty[]=0&status[]=solved&category[]=Heap&sortBy=submissions

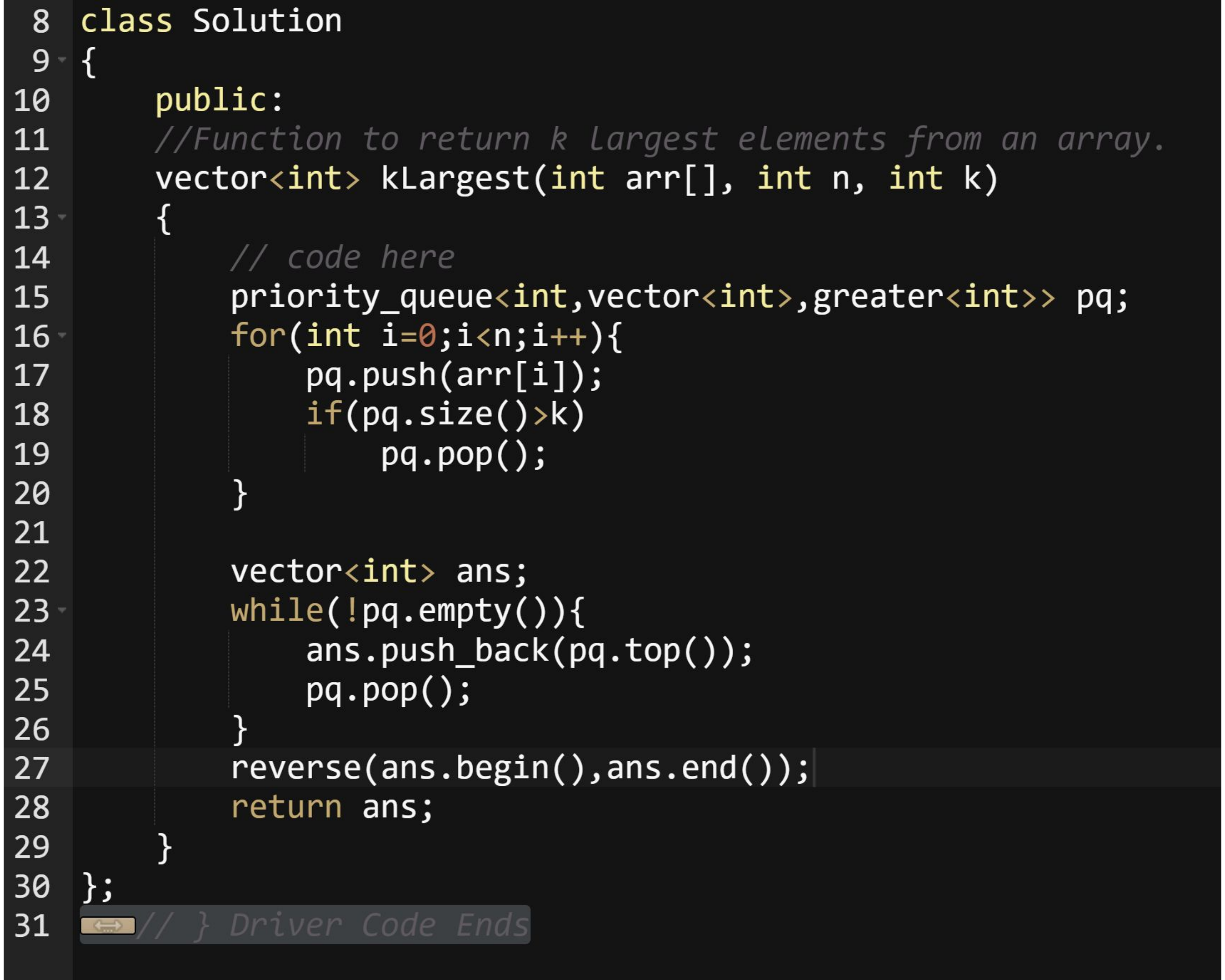


```cpp
class Solution{
public:
    int minValue(string s, int k){
        // code here
        vector<int> freq(26,0);
        for(auto i:s)
            freq[i-'a']++;
        priority_queue<int> pq;
        for(int i=0;i<26;i++)
            if(freq[i]!=0)
                pq.push(freq[i]);
        while(k--){
            int a = pq.top();
            pq.pop();
            a--;
            if(a>0)
                pq.push(a);
        }
        int ans = 0;
        while(!pq.empty()){
            int a = pq.top();
            ans+=a*a;
            pq.pop();
        }
        return ans;
    }
};
// } Driver Code Ends
```

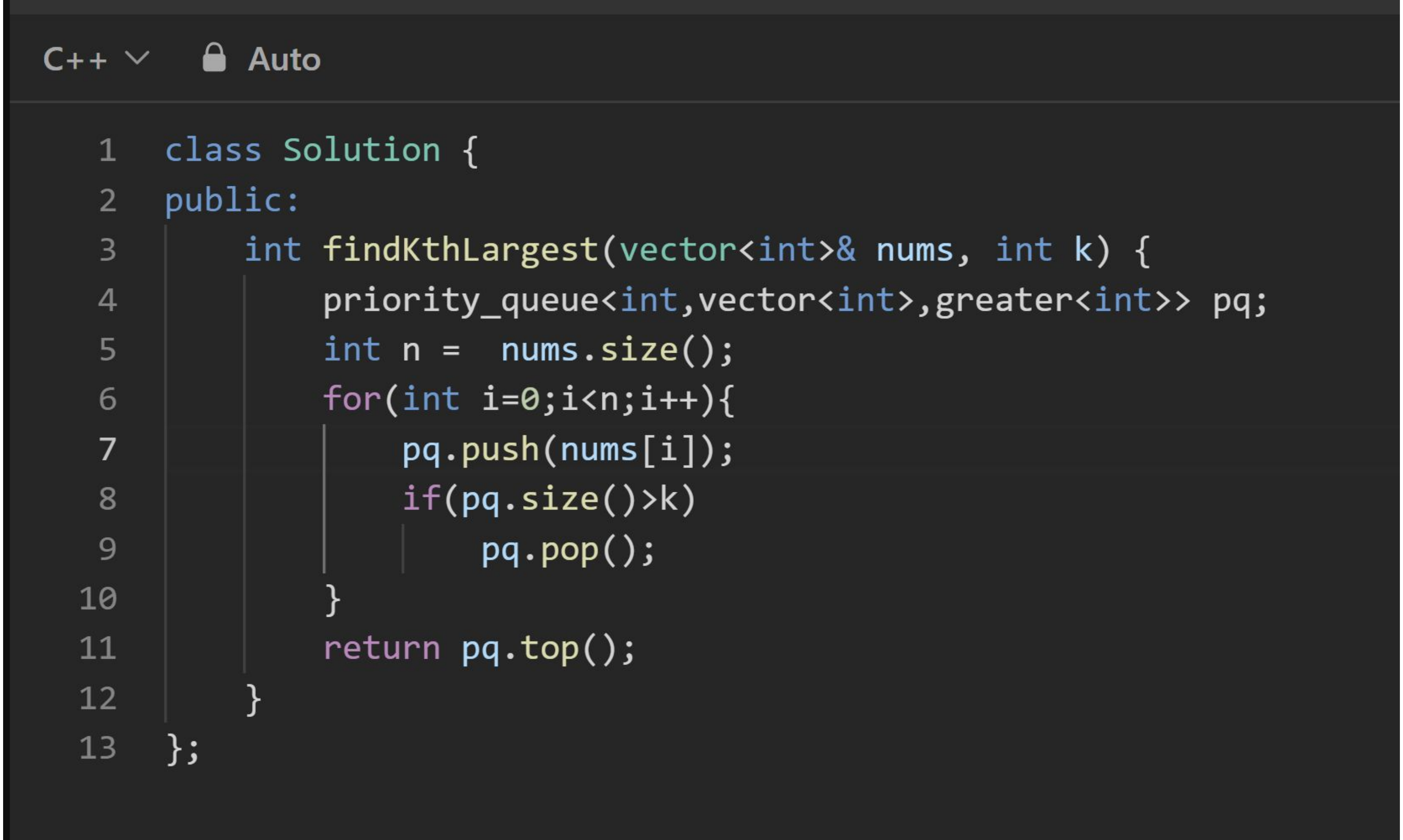
Question

Kth largest element in array

- https://leetcode.com/problems/kth-largest-element-in-an-array/
- https://practice.geeksforgeeks.org/problems/k-largest-elements3736/1/?page=1&difficulty[]=-1&status[]=solved&category[]=Heap&sortBy=submissions



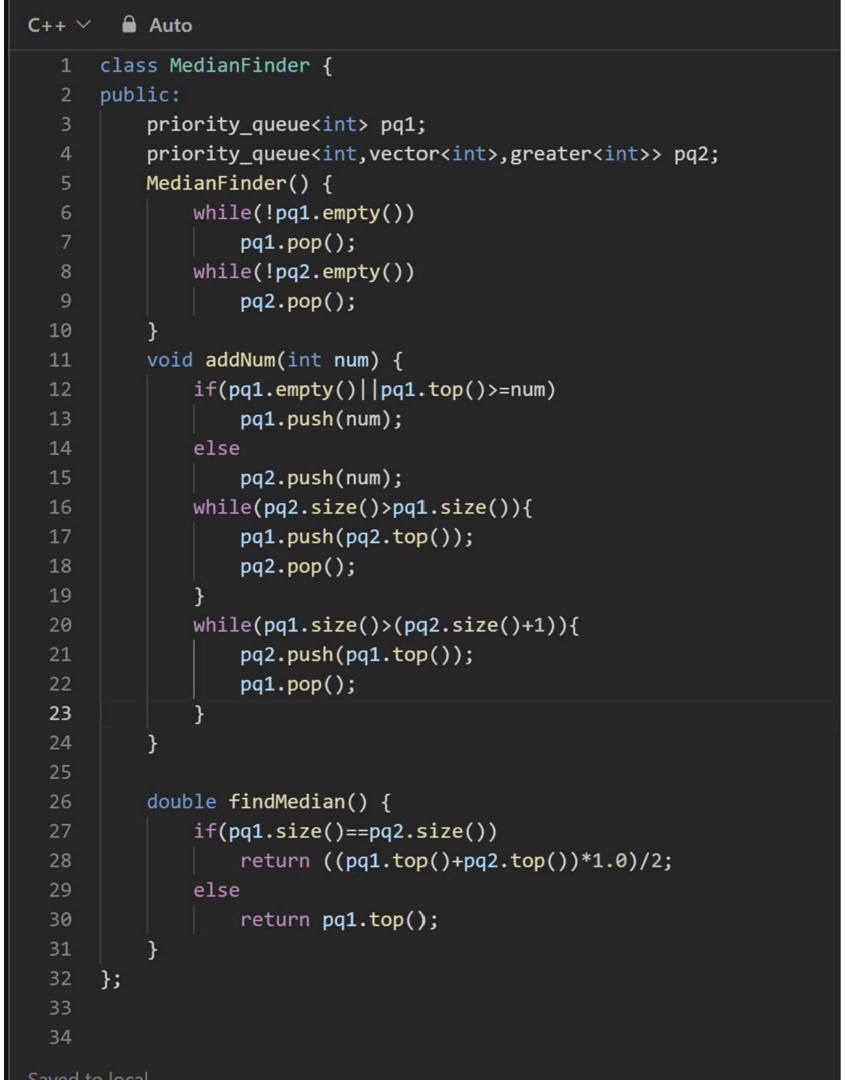
```cpp
class Solution
{
    public:
    //Function to return k largest elements from an array.
    vector<int> kLargest(int arr[], int n, int k)
    {
        // code here
        priority_queue<int,vector<int>,greater<int>> pq;
        for(int i=0;i<n;i++){
            pq.push(arr[i]);
            if(pq.size()>k)
                pq.pop();
        }

        vector<int> ans;
        while(!pq.empty()){
            ans.push_back(pq.top());
            pq.pop();
        }
        reverse(ans.begin(),ans.end());
        return ans;
    }
};
// } Driver Code Ends
```



```cpp
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        priority_queue<int,vector<int>,greater<int>> pq;
        int n =  nums.size();
        for(int i=0;i<n;i++){
            pq.push(nums[i]);
            if(pq.size()>k)
                pq.pop();
        }
        return pq.top();
    }
};
```

Question

https://leetcode.com/problems/find-median-from-data-stream/description/



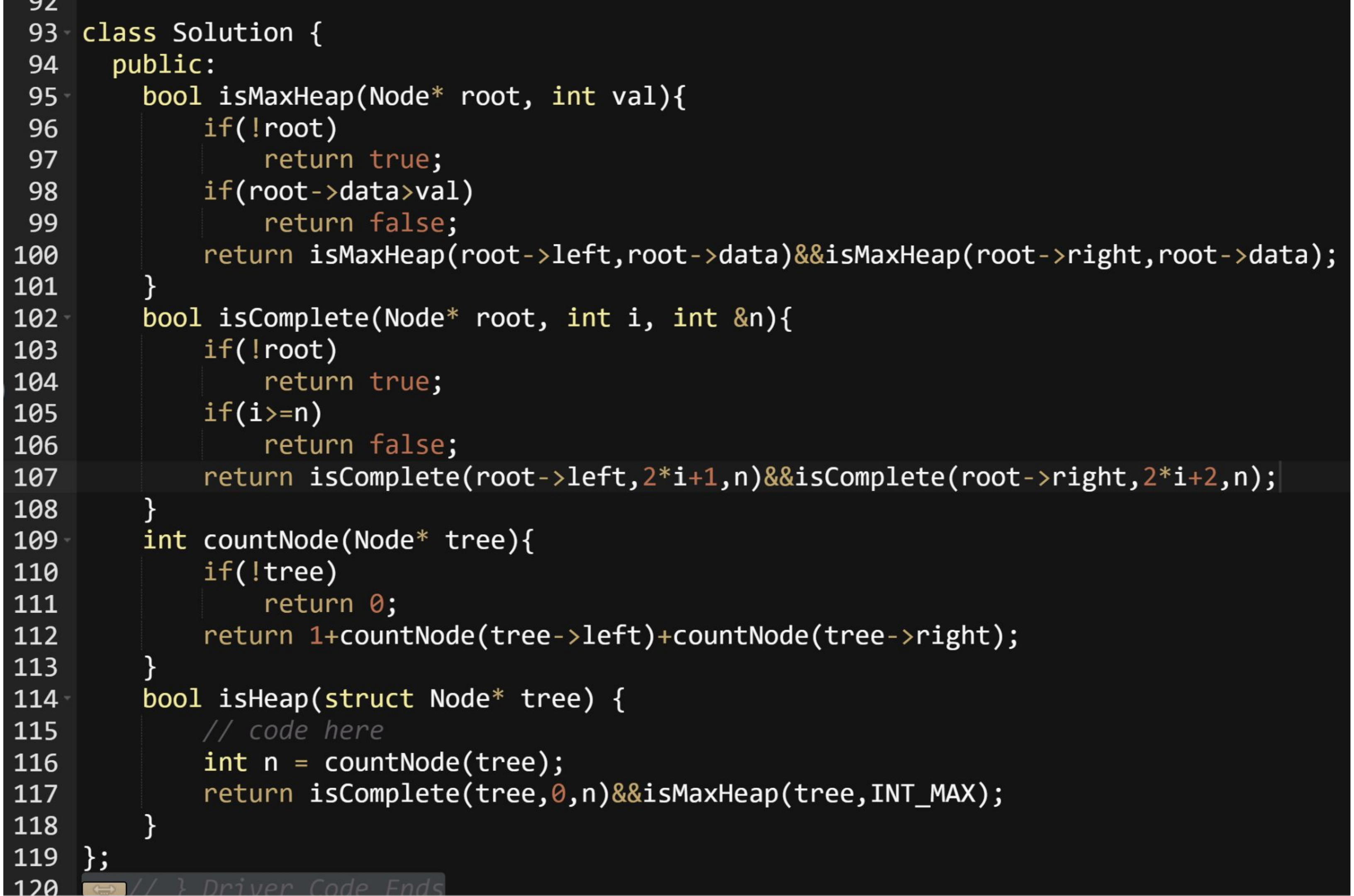
```cpp
class MedianFinder {
public:
    priority_queue<int> pq1;
    priority_queue<int,vector<int>,greater<int>> pq2;
    MedianFinder() {
        while(!pq1.empty())
            pq1.pop();
        while(!pq2.empty())
            pq2.pop();
    }
    void addNum(int num) {
        if(pq1.empty()||pq1.top()>=num)
            pq1.push(num);
        else
            pq2.push(num);
        while(pq2.size()>pq1.size()){
            pq1.push(pq2.top());
            pq2.pop();
        }
        while(pq1.size()>(pq2.size()+1)){
            pq2.push(pq1.top());
            pq1.pop();
        }
    }

    double findMedian() {
        if(pq1.size()==pq2.size())
            return ((pq1.top()+pq2.top())*1.0)/2;
        else
            return pq1.top();
    }
};
```

Question

Check if binary tree is heap

Link :

https://practice.geeksforgeeks.org/problems/is-binary-tree-heap/1?page=1&difficulty%5B%5D=0&status%5B%5D=solved&category%5B%5D=Heap&sortBy=submissions



```
class Solution {
  public:
    bool isMaxHeap(Node* root, int val){
        if(!root)
            return true;
        if(root->data>val)
            return false;
        return isMaxHeap(root->left,root->data)&&isMaxHeap(root->right,root->data);
    }
    bool isComplete(Node* root, int i, int &n){
        if(!root)
            return true;
        if(i>=n)
            return false;
        return isComplete(root->left,2*i+1,n)&&isComplete(root->right,2*i+2,n);
    }
    int countNode(Node* tree){
        if(!tree)
            return 0;
        return 1+countNode(tree->left)+countNode(tree->right);
    }
    bool isHeap(struct Node* tree) {
        // code here
        int n = countNode(tree);
        return isComplete(tree,0,n)&&isMaxHeap(tree,INT_MAX);
    }
};
// } Driver Code Ends
```