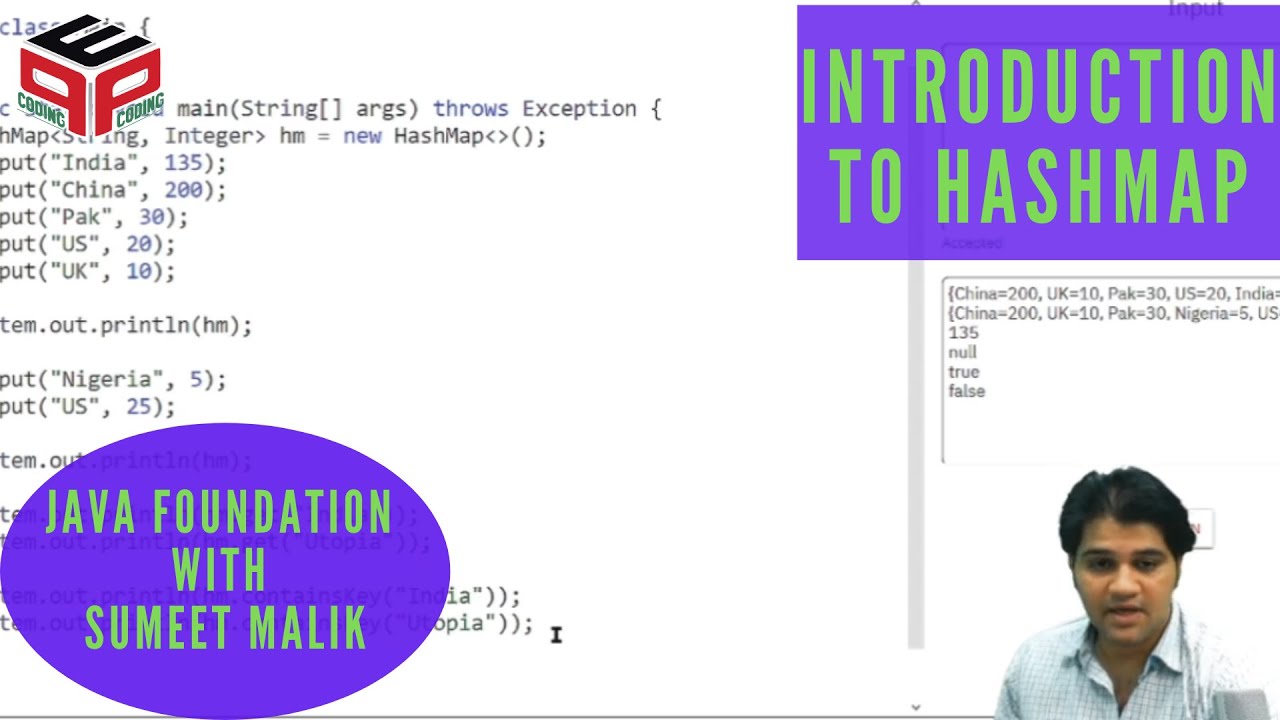
**Introduction To Hashmap[](https://www.youtube.com/watch?v=ukTRZD-WJH4)[](https://www.youtube.com/watch?v=m22CHxE_YmI)**



Highest Frequency Character

Easy

1. You are given a string str.

2. You are required to find the character with maximum frequency.

**Constraints**

0 < str.length() <= 100

There will be a single character with highest frequency

**Format**

**Input**

A string str

**Output**

The character with highest frequency

**Example**

**Sample Input**

zmszeqxllzvheqwrofgcuntypejcxovtaqbnqyqlmrwitc

**Sample Output**

q

#include <bits/stdc++.h>

using namespace std;

int main() {

string str;

cin >> str;

// write your code here

unordered\_map<char, int> occ;

for(int i {}; i< str.size(); i++){

occ[str[i]]++;

}

int max = INT\_MIN;

char ans;

for(pair<char, int> p : occ){

if(p.second > max ){

max = p.second;

ans = p.first;

}

}

cout<<ans<<endl;

return 0;

}

Get Common Elements - 1

Easy

1. You are given a number n1, representing the size of array a1.

2. You are given n1 numbers, representing elements of array a1.

3. You are given a number n2, representing the size of array a2.

4. You are given n2 numbers, representing elements of array a2.

5. You are required to print all elements of a2 which are also present in a1 (in order of their occurence in a2). Make sure to not print duplicates (a2 may have same value present many times).

**Constraints**

1 <= n1, n2 <= 100

0 <= a1[i], a2[i] < 10

Time complexity should be O(n)

**Format**

**Input**

A number n1

n1 number of elements line separated

A number n2

n2 number of elements line separated

**Output**

All relevant elements of a2 in separate lines (no duplicacy)

**Example**

**Sample Input**

9

5

5

9

8

5

5

8

0

3

18

9

7

1

0

3

6

5

9

1

1

8

0

2

4

2

9

1

5

**Sample Output**

9

0

3

5

8

#include <iostream>

#include <unordered\_map>

using namespace std;

int main() {

int n1,n2;

cin >> n1;

int arr1[n1];

unordered\_map <int,int> mp;

for (int i = 0; i < n1; i++) {

cin>>arr1[i];

mp[arr1[i]]++;

}

cin >> n2;

int arr2[n2];

for (int i = 0; i < n2; i++) {

cin >> arr2[i];

// write your code here

if(mp.find(arr2[i]) != mp.end() ){

cout<<arr2[i]<<endl;

mp.erase(arr2[i]);

}

}

}

Get Common Elements - 2

Easy

1. You are given a number n1, representing the size of array a1. 2. You are given n1 numbers, representing elements of array a1. 3. You are given a number n2, representing the size of array a2. 4. You are given n2 numbers, representing elements of array a2. 5. You are required to find the intersection of a1 and a2. To get an idea check the example below: if a1 -> 1 1 2 2 2 3 5 and a2 -> 1 1 1 2 2 4 5 intersection is -> 1 1 2 2 5 Note -> Don't assume the arrays to be sorted. Check out the question video.

**Constraints**

1 <= n1, n2 <= 100 0 <= a1[i], a2[i] < 10 Time complexity should be O(n)

**Format**

**Input**

A number n1 n1 number of elements line separated A number n2 n2 number of elements line separated

**Output**

All relevant elements of intersection in separate lines The elements of intersection should be printed in order of their occurence in a2.

**Example**

**Sample Input**

7

1

1

2

2

2

3

5

7

1

1

1

2

2

4

5

**Sample Output**

1

1

2

2

5

#include <iostream>

#include <unordered\_map>

using namespace std;

int main() {

//write your code here

int n1{};

cin >> n1;

unordered\_map <int,int> um;

for(int i {};i < n1;i++){

int a ;

cin>> a;

um[a]++;

}

int n2{};

cin>>n2;

for(int i{} ; i< n2;i++){

int a{};

cin>> a;

if(um.find(a) != um.end() && um[a] > 0){

cout<<a<<endl;

um[a]--;

}

}

return 0;

}

Longest Consecutive Sequence Of Elements

Hard

1. You are given a number n, representing the size of array a.

2. You are given n numbers, representing elements of array a.

3. You are required to print the longest sequence of consecutive elements in the array (ignoring duplicates).

Note -> In case there are two sequences of equal length (and they are also the longest), then print the one for which the starting point of which occurs first in the array.

**Constraints**

1 <= n <= 30

0 <= n1, n2, .. n elements <= 15

**Format**

**Input**

A number n

n1

n2

.. n number of elements

**Output**

Elements of longest sequence of consecutive elements of array in separate lines (ignoring duplicates)

**Example**

**Sample Input**

17

12

5

1

2

10

2

13

7

11

8

9

11

8

9

5

6

11

**Sample Output**

5

6

7

8

9

10

11

12

13

#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

void longestConsecutive(vector<int> &num) {

//write your code here

//on leet code my solution was more efficient in time complexity but less in space

unordered\_map <int, bool> um;

for(int n : num){

um[n] = true;

}

for(int n: num){

if(um.find(n-1) != um.end()) {

um[n] = false; // it is not the starting of any sequence

}

}

int max\_start\_ele{};

int max\_length{};

for(int n: num){

if(um[n] == true) {

int t\_length{1};

while(um.find(n + t\_length) != um.end()){

t\_length++;

}

// cout<<n<<"--"<<t\_length<<endl;

if(t\_length > max\_length){

max\_length = t\_length;

max\_start\_ele = n;

}

}

}

int a = max\_start\_ele;

// cout<<max\_start\_ele<<"--"<<max\_length<<endl;

for(int i {} ;i < max\_length ;i++){

cout<<a<<endl;

a++;

}

}

int main()

{

vector<int>arr;

int n;

cin >> n;

for (int i = 0 ; i < n; i++) {

int data;

cin >> data;

arr.push\_back(data);

}

longestConsecutive(arr);

return 0;

}

/\*

my solution

#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

int length\_from\_this\_element(int ele,unordered\_map <int, int> &um){

if(um.find(ele) == um.end()){

return 0;

}

if(um[ele] > 0){

return um[ele];

}

int ahead\_lenght = length\_from\_this\_element(ele+1,um);

um[ele] = ahead\_lenght + 1;

return um[ele];

}

void longestConsecutive(vector<int> &num) {

//write your code here

int start\_ele {};

// int start\_ele\_ind {};

int max\_conse\_length{};

unordered\_map <int,int> um;

for(int i{} ;i < num.size() ; i++ ) {

um[num[i]] = 0;

}

int this\_length {};

for(int i{} ;i < num.size() ; i++ ) {

if(um[num[i]] == 0){

this\_length = length\_from\_this\_element(num[i], um);

}

if(this\_length > max\_conse\_length){

start\_ele = num[i];

// start\_ele\_ind = i;

max\_conse\_length = this\_length;

}

}

for(int i {};i < max\_conse\_length ;i++){

cout<<start\_ele<<endl;

start\_ele++;

}

}

int main()

{

vector<int>arr;

int n;

cin >> n;

for (int i = 0 ; i < n; i++) {

int data;

cin >> data;

arr.push\_back(data);

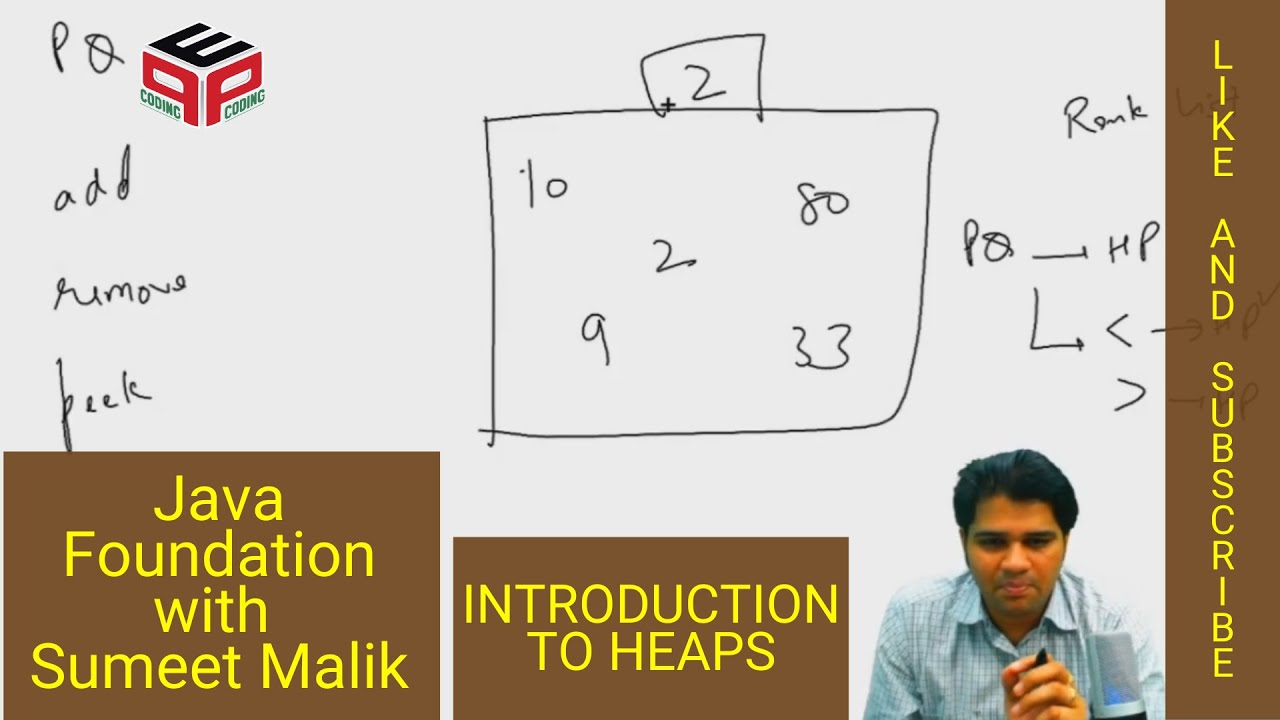
}

longestConsecutive(arr);

return 0;

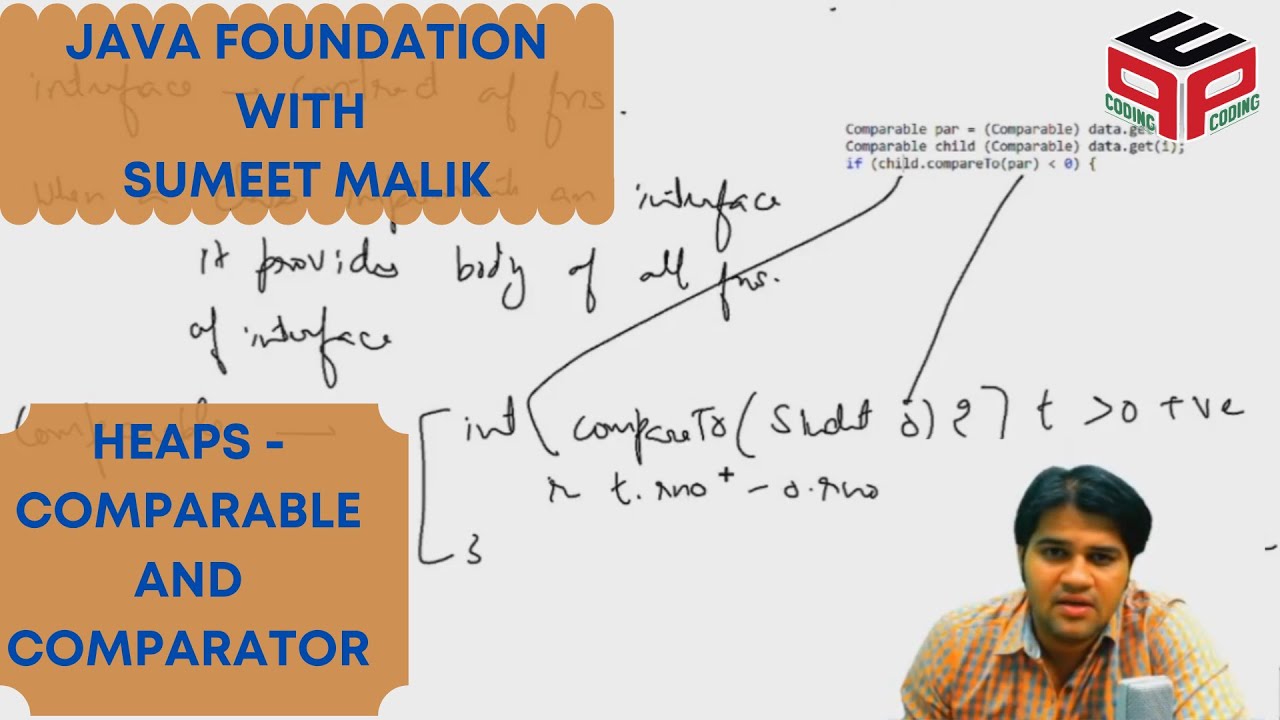
}

\*/

**Introduction To Heap[](https://www.youtube.com/watch?v=eSwIfy2xJ1g)[](https://www.youtube.com/watch?v=BGuv_FFjRts)**

**Efficient Heap Constructor[](https://www.youtube.com/watch?v=VFLNBh0bK2A)[](https://www.youtube.com/watch?v=CJQ5It9HJ8s)**

**Heap - Comparable V/S Comparator**

[](https://www.youtube.com/watch?v=DE835GvxZMQ)

K Largest Elements

Easy

1. You will be given an Array with its length

2. You will also be given an integral value k

3. You need to find k largest elements from the given array

4. Input is handled for you

5. It is a functional problem ,please do not modify main()

**Constraints**

1 <= N <= 100

K <= N

-1000 <= C[i] <= 1000

**Format**

**Input**

Input is handled for you

**Output**

Output MUST be in descending order

**Example**

**Sample Input**

8

44 -5 -2 41 12 19 21 -6

2

**Sample Output**

44 41

#include<iostream>

#include<vector>

#include<string>

#include<queue>

using namespace std;

// -----------------------------------------------------

// This is a functional problem. Only this function has to be written.

// This function takes as input an array,n length of array and k.

// It should print required output.

void solve(int n,vector<int>& arr,int k){

//Write your code here

priority\_queue<int,vector<int>,greater<int>> pq;

for(int a:arr){

if(pq.size() < k){

pq.push(a);

}else{

if(a > pq.top()){

pq.pop();

pq.push(a);

}

}

}

string ans{""};

while(pq.size() >0){

ans = to\_string(pq.top()) + " "+ans;

pq.pop();

}

cout<<ans<<endl;

}

int main(int argc,char\*\* argv){

int n;

cin>>n;

vector<int> v(n);

for(int i=0;i<n;i++)

cin>>v[i];

int k;

cin>>k;

solve(n,v,k);

}

Sort K-sorted Array

Easy

1. You are given a number n, representing the size of array a.

2. You are given n numbers, representing elements of array a.

3. The array is nearly sorted. Every element is at-max displaced k spots left or right to it's position in the sorted array. Hence it is being called k-sorted array.

4. You are required to sort and print the sorted array.

Note -> You can use at-max k extra space and nlogk time complexity.

**Constraints**

1 <= n <= 30

0 <= n1, n2, .. n elements <= 100

0 < k <= n

**Format**

**Input**

Input is managed for you

**Output**

Print the elements of sorted array in separate lines.

**Example**

**Sample Input**

9

3

2

4

1

6

5

7

9

8

3

**Sample Output**

1

2

3

4

5

6

7

8

9

#include <bits/stdc++.h>

using namespace std;

// #include <iostream>

// #include <vector>

// #include <queue>

// using namespace std;

void sortK(int arr[], int n, int k){

//write your code here

priority\_queue<int,vector<int>,greater<int>> pq;

// cout<<"------------------------------"<<endl;

//queue size k

// for(int i{} ;i < n ;i++) {

// if(pq.size() < k){

// pq.push(arr[i]);

// }else{

// if(arr[i] > pq.top()){

// cout<<pq.top()<<endl;

// pq.pop();

// pq.push(arr[i]);

// }else{

// cout<<arr[i]<<endl;

// }

// }

// }

//

//queue size k+1

for(int i{} ; i <= k ;i++) {

pq.push(arr[i]);

}

for(int i{k+1} ;i< n;i++){

cout<<pq.top()<<endl;

pq.pop();

pq.push(arr[i]);

}

while(pq.size() > 0) {

cout<<pq.top()<<endl;

pq.pop();

}

}

int main()

{

int n;

cin>>n;

int arr[n];

for(int i = 0 ; i<n ; i++){

int data;

cin>>data;

arr[i]=data;

}

int k;

cin>>k;

sortK(arr, n, k);

return 0;

}

Median Priority Queue

Hard

1. You are required to complete the code of our MedianPriorityQueue class. The class should mimic the behaviour of a PriorityQueue and give highest priority to median of it's data.

2. Here is the list of functions that you are supposed to complete

2.1. add -> Should accept new data.

2.2. remove -> Should remove and return median value, if available or print "Underflow" otherwise and return -1

2.3. peek -> Should return median value, if available or print "Underflow" otherwise and return -1

2.4. size -> Should return the number of elements available

3. Input and Output is managed for you.

Note -> If there are even number of elements in the MedianPriorityQueue, consider the smaller median as median value.

**Constraints**

None

**Format**

**Input**

Input is managed for you

**Output**

Output is managed for you

**Example**

**Sample Input**

add 10

add 20

add 30

add 40

peek

add 50

peek

remove

peek

remove

peek

remove

peek

remove

peek

quit

**Sample Output**

20

30

30

20

20

40

40

10

10

50

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

class MedianPriorityQueue {

public:

priority\_queue <int> left;

priority\_queue <int, vector<int>, greater<int>> right;

void push(int val) {

//write your code here

//can see sir solution in video

//my solution

if(left.size() == right.size()) {

if(left.size() == 0 || val < left.top()){

left.push(val);

}else{

right.push(val);

left.push(right.top());

right.pop();

}

}else{

if(right.size() == 0 || val < right.top()){

left.push(val);

right.push(left.top());

left.pop();

}else{

right.push(val);

}

}

}

int pop() {

//write your code here

if(left.size() == 0){

cout<<"Underflow"<<endl;

return -1;

}

int median = left.top();

left.pop();

if(left.size() < right.size()){

left.push(right.top());

right.pop();

}

return median;

}

int top() {

//write your code here

if(left.size() == 0){

cout<<"Underflow"<<endl;

return -1;

}

return left.top();;

}

int size() {

//write your code here

return left.size() + right.size();

}

};

int main() {

MedianPriorityQueue pq;

string str;

cin >> str;

while(str!="quit"){

if(str=="add"){

int val;

cin >> val;

pq.push(val);

}

else if(str=="remove"){

int val = pq.pop();

if(val != -1) {

cout<<val<<endl;

}

}

else if(str=="peek"){

int val = pq.top();

if(val != -1) {

cout<<val<<endl;

}

}

else if(str=="size"){

cout<<pq.size()<<endl;

}

cin >> str;

}

return 0;

}

Merge K Sorted Lists

Hard

1. You are given a list of lists, where each list is sorted.

2. You are required to complete the body of mergeKSortedLists function. The function is expected to merge k sorted lists to create one sorted list.

**Constraints**

Space complextiy = O(k)

Time complexity = nlogk

where k is the number of lists and n is number of elements across all lists.

**Format**

**Input**

Input is managed for you

**Output**

Output is managed for you

**Example**

**Sample Input**

4

5

10 20 30 40 50

7

5 7 9 11 19 55 57

3

1 2 3

2

32 39

**Sample Output**

1 2 3 5 7 9 10 11 19 20 30 32 39 40 50 55 57

#include<iostream>

#include<queue>

#include<unordered\_map>

#include<vector>

using namespace std;

class Pair{

public:

int val;

int vi;

int di;

Pair(int val, int vi,int di) {

this-> val = val;

this-> vi = vi;

this-> di = di;

}

};

struct my\_comp{

bool operator()(const Pair & a, const Pair & b){

return a.val > b.val;

}

};

vector<int> mergeKSortedLists(vector<vector<int>>&lists) {

vector<int> rv;

//write your code here

priority\_queue<Pair,vector<Pair>, my\_comp> pq;

for(int i{};i<lists.size() ;i++) {

Pair pr (lists[i][0],i,0);

pq.push(pr);

}

while(pq.size() > 0) {

Pair a = pq.top();

pq.pop();

rv.push\_back(a.val);

a.di++;

if(a.di < lists[a.vi].size()){

a.val = lists[a.vi][a.di];

pq.push(a);

}

}

// without using pair class in priority\_queue

// priority\_queue<int,vector<int>, greater<int>> pq;

// unordered\_map<int,int> um;

// for(int i{};i<lists.size() ;i++) {

// um[lists[i][0]] = i;

// pq.push(lists[i][0]);

// }

// vector<int> index (lists.size(),1);

// cout<<"11111"<<endl;

// // // cout<<"--"<<pq.top()<<endl;

// // int t = 5;

// while(pq.size() > 0) {

// cout<<"--"<<pq.top()<<endl;

// int t = pq.top();pq.pop();

// rv.push\_back(t);

// int v\_index = um[t];

// if(index[v\_index] < lists[v\_index].size() ){

// um[lists[v\_index][index[v\_index]]] =um[t];

// um.erase(t);

// pq.push(lists[v\_index][index[v\_index]]);

// index[v\_index]++;

// }

// }

return rv;

}

int main() {

int k;

cin >> k;

vector<vector<int>>lists;

for (int i = 0; i < k; i++) {

vector<int>list;

int n;

cin >> n;

for (int j = 0; j < n; j++) {

int data;

cin >> data;

list.push\_back(data);

}

lists.push\_back(list);

}

vector<int> mlist = mergeKSortedLists(lists);

for (int val : mlist) {

cout << val << " ";

}

cout <<endl;

return 0;

}

Write Priority Queue Using Heap

Easy

1. You are required to complete the code of our Priority Queue class using the heap data structure. Please watch the question video carefully. The theoretical details of required functionality is explained in detail there. Implement the functions to achieve what is explained in the theoretical discussion in question video.

2. Here is the list of functions that you are supposed to complete:

2.1. add -> Should accept new data.

2.2. remove -> Should remove and return smallest value, if available or print

"Underflow" otherwise and return -1.

2.3. peek -> Should return smallest value, if available or print "Underflow"

otherwise and return -1.

2.4. size -> Should return the number of elements available.

3. Input and Output is managed for you.

**Constraints**

None

**Format**

**Input**

Input is managed for you

**Output**

Output is managed for you

**Example**

**Sample Input**

add 10

add 20

add 30

add 40

peek

add 50

peek

remove

peek

remove

peek

remove

peek

remove

peek

quit

**Sample Output**

10

10

10

20

20

30

30

40

40

50

#include<bits/stdc++.h>

using namespace std;

// #include<iostream>

// #include<string>

// #include<vector>

// using namespace std;

vector<int> data;

int \_size() {

//write your code here

return data.size();//-----------------------------------------

}

void swap(int a , int b){

int temp = data[a];

data[a] = data[b];

data[b] = temp;

}

void upheapify(int index) {

if(index == 0 ) {

return ;

}

int pi = (index - 1) / 2;

if(data[index] < data[pi]){

//swap

swap(index, pi);

//upheapify

upheapify(pi);

}

}

void add(int val) {

// write your code here

data.push\_back(val);

upheapify(data.size() - 1);

}

void down\_heapify(int p\_index) {

int min = p\_index;

int li = (2 \* p\_index) +1;

if(li < data.size() && data[li] < data[min]){

min = li;

}

int ri = (2 \* p\_index) +2;

if(ri < data.size() && data[ri] < data[min]) {

min = ri;

}

if(min != p\_index){

swap(p\_index, min);

down\_heapify(min);

}

}

int \_remove() {

//write your code here

if(data.size() == 0){

cout<<"Underflow"<<endl;

return -1;

}

//swap 1st and last

swap(0,data.size() -1);

//remove last

int ans = data[data.size() -1];

data.pop\_back();

//down\_heapify

down\_heapify(0);

return ans;

}

int peek() {

//write your code here

if(data.size() == 0){

cout<<"Underflow"<<endl;

return -1;

}

return data[0];

}

int main(){

while(1){

string str;

getline(cin,str);

if(str[0]=='a'){

string num=str.substr(4,str.length());

int val=stoi(num);

add(val);

}else if(str[0]=='r'){

int val=\_remove();

if(val!=-1){

cout<<val<<endl;

}

}else if(str[0]=='p'){

int val=peek();

if(val!=-1){

cout<<val<<endl;

}

}else break;

}

}

Write Hashmap

Easy

1. You are required to complete the code of our Hashmap class. Please watch the question video carefully. The theoretical details of required functionality is explained in detail there. Implement the functions to achieve what is explained in the theoretical discussion in question video.

2. Input and Output is managed for you.

**Constraints**

None

**Format**

**Input**

Input is managed for you

**Output**

Output is managed for you

**Example**

**Sample Input**

put India 135

put Aus 5

put Canada 3

display

get China

remove Aus

get Aus

containsKey US

put US 10

put UK 20

remove US

containsKey US

put Pak 80

put China 200

display

put Utopia 0

display

put Nigeria 3

display

put India 138

display

put Sweden 1

display

put finland 20

display

quit

**Sample Output**

Display Begins

Bucket0 .

Bucket1 .

Bucket2 Canada@3 .

Bucket3 India@135 Aus@5 .

Display Ends

null

5

null

false

10

false

Display Begins

Bucket0 .

Bucket1 .

Bucket2 Canada@3 UK@20 Pak@80 .

Bucket3 India@135 China@200 .

Display Ends

Display Begins

Bucket0 Utopia@0 .

Bucket1 .

Bucket2 Canada@3 UK@20 Pak@80 .

Bucket3 India@135 China@200 .

Display Ends

Display Begins

Bucket0 Utopia@0 .

Bucket1 .

Bucket2 Canada@3 UK@20 Pak@80 .

Bucket3 India@135 China@200 Nigeria@3 .

Display Ends

Display Begins

Bucket0 Utopia@0 .

Bucket1 .

Bucket2 Canada@3 UK@20 Pak@80 .

Bucket3 India@138 China@200 Nigeria@3 .

Display Ends

Display Begins

Bucket0 Utopia@0 Sweden@1 .

Bucket1 .

Bucket2 Canada@3 UK@20 Pak@80 .

Bucket3 India@138 China@200 Nigeria@3 .

Display Ends

Display Begins

Bucket0 Utopia@0 .

Bucket1 .

Bucket2 Pak@80 finland@20 .

Bucket3 .

Bucket4 Sweden@1 .

Bucket5 .

Bucket6 Canada@3 UK@20 .

Bucket7 India@138 China@200 Nigeria@3 .

Display Ends

//no submission

I think there is little bit in base case (vscode)