**Assignment - 43 A Job Ready Bootcamp in C++, DSA and IOT**

**priority\_queue**

1. Write a c++ program, to demonstrate priority queue.

Sol – 1.

#include<bits/stdc++.h>

using namespace std;

int main()

{

priority\_queue<int>q;

q.push(23);

q.push(12);

q.push(64);

q.push(40);

cout<<"Size : "<<q.size()<<endl;

cout<<"Top : "<<q.top()<<endl;

cout<<"Priority Queue is"<<endl;

while(!q.empty())

{

cout<<q.top()<<" ";

q.pop();

}

cout<<endl;

return 0;

}

2. Implement different operations on priority queue .i.e. adding element, removing

element, size of priority queue, and print it.

Sol – 2.

Same as 1

3. Write a c++ program, to demonstrate priority queue having a min element at top.

Sol – 3.

#include<bits/stdc++.h>

using namespace std;

int main()

{

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

q.push(23);

q.push(12);

q.push(64);

q.push(40);

cout<<"Size : "<<q.size()<<endl;

cout<<"Top : "<<q.top()<<endl;

cout<<"Priority Queue is"<<endl;

while(!q.empty())

{

cout<<q.top()<<" ";

q.pop();

}

cout<<endl;

return 0;

}

4. Write a c++ program, to swap the elements of two priority queues of int type.

Sol – 4.

#include<bits/stdc++.h>

using namespace std;

void showque(priority\_queue<int>q)

{

while(!q.empty())

{

cout<<q.top()<<" ";

q.pop();

}

cout<<endl;

}

int main()

{

priority\_queue<int>q;

q.push(23);

q.push(12);

q.push(64);

q.push(40);

priority\_queue<int>q2;

q2.push(4);

q2.push(66);

q2.push(27);

q2.push(11);

cout<<"Before Swapping"<<endl;

cout<<"Priority Queue 1 is"<<endl;

showque(q);

cout<<"Priority Queue 2 is"<<endl;

showque(q2);

q.swap(q2);

cout<<"After Swapping"<<endl;

cout<<"Priority Queue 1 is"<<endl;

showque(q);

cout<<"Priority Queue 2 is"<<endl;

showque(q2);

return 0;

}

5. Write a c++ program, to show that priority\_queue is by default a Max Heap.

Note:

If elements are printed in descending order, then we have a max heap.

Sol – 5.

Same as 1

6. Write a c++ program, to use priority\_queue to implement min heap.

Sol – 6.

Same as 3

7. Given two sorted arrays A[] and B[] of sizes N and M respectively, the task is to

merge them in a sorted manner using priority\_queue.

Example:

Input: A[] = { 5, 6, 8 }, B[] = { 4, 7, 8 }

Output: 4 5 6 7 8 8

Sol – 7.

#include<bits/stdc++.h>

using namespace std;

int main()

{

int a[]={5,8,3};

int b[]={4,9,7};

int m=sizeof(a)/sizeof(a[0]);

int n=sizeof(b)/sizeof(b[0]);

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

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

{

pq.push(a[i]);

}

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

{

pq.push(b[i]);

}

while(!pq.empty())

{

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

pq.pop();

}

cout<<endl;

return 0;

}

8. Given an array arr[] of N elements, the task is to perform using priority\_queue and

the following operation:

● Pick the two largest element from the array and remove these element. If the

elements are unequal then insert the absolute difference of the elements into the

array.

● Perform the above operations until the array has 1 or no element in it. If the array has

only one element left then print that element, else print “-1”.

Example:

Input: arr[] = { 3, 5, 2, 7 }

Output: 1

Explanation:

The two largest elements are 7 and 5. Discard them. Since both are not equal, insert

7 – 5 = 2 into the array. Hence, arr[] = { 3, 2, 2 }

The two largest elements are 3 and 2. Discard them. Since both are not equal, insert

3 – 2 = 1 into the array. Hence, arr[] = { 1, 2 }

The two largest elements are 2 and 1. Discard them. Since both are not equal, insert

2 – 1 = 1 into the array. Hence, arr[] = { 1 }

The only element left is 1. Print the value of the only element left.

Sol – 8.

#include<bits/stdc++.h>

using namespace std;

int main()

{

int a[]={3,5,2,7};

int n=sizeof(a)/sizeof(a[0]);

priority\_queue<int>pq;

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

{

pq.push(a[i]);

}

int p,q;

while(pq.size()>1)

{

p=pq.top();

pq.pop();

q=pq.top();

pq.pop();

if(p!=q)

pq.push(p-q);

}

if(pq.size()==1)

cout<<"Element left : "<<pq.top();

else

cout<<"-1"<<endl;

return 0;

}

9. Given three arrays X[], Y[], and Z[] each consisting of N integers, the task is to find

the maximum number of triplets (X[i], Y[i], Z[i]) such that (X[i] < Y[i] < Z[i]) for any

permutation of the three arrays using priority\_queue

Input: X ={9, 6, 14, 1, 8}, Y = {2, 10, 3, 12, 11}, Z = {15, 13, 5, 7, 4}

Output: 3

Explanation:

After rearranging the arrays X[], Y[] and Z[] as {1, 6, 8, 9, 14}, {3, 2, 10, 12, 11}, and

{4, 7, 15, 13, 5} respectively. The increasing triplets are {1, 3, 4}, {8, 10, 15} and {9,

12, 13}.

Therefore, the total count of such triplets is 3.

Sol – 9.

#include<bits/stdc++.h>

using namespace std;

int counttriplet(int X[],int y[],int z[],int n)

{

sort(X,X+n);

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

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

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

{

Y.push(y[i]);

Z.push(z[i]);

}

int ans=0;

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

{

int x=X[i];

while(!Y.empty()&&Y.top()<=x)

Y.pop();

if(Y.empty())

break;

int y=Y.top();

Y.pop();

while(!Z.empty()&&Z.top()<=y)

Z.pop();

if(Z.empty())

break;

int z=Z.top();

Z.pop();

++ans;

}

return ans;

}

int main()

{

int X[]={9,6,14,1,8};

int Y[]={2,10,3,12,11};

int Z[]={15,13,5,7,4};

int n=sizeof(X)/sizeof(X[0]);

cout<<counttriplet(X,Y,Z,n);

return 0;

}

10. Given an array arr[] of size N and a number K, the task is to find the length of the

smallest subsequence such that the sum of the subsequence is greater than or equal

to number K do it using priority\_queue.

Example:

Input: arr[] = {2, 3, 1, 5, 6, 3, 7, 9, 14, 10, 2, 5}, K = 35

Output: 4

Smallest subsequence with the sum greater than or equal to the given sum K is {7, 9,

14, 10}

Input: arr[] = {1, 2, 2, 2, 3, 4, 5, 4, 7, 6, 5, 12}, K = 70

Output:-1

Subsequence with sum greater than equal to the given sum is not possible.

Sol – 10.

#include<bits/stdc++.h>

using namespace std;

int count(int x[],int n,int k)

{

priority\_queue<int>q;

int sum=0,ans=0;

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

{

q.push(x[i]);

}

while(sum<k)

{

if(q.empty())

{

ans = -1;

break;

}

sum=sum+q.top();

q.pop();

ans++;

}

return ans;

}

int main()

{

int X[]={1, 2, 2, 2, 3, 4, 5, 4, 7, 6, 5, 12};

int k=70;

int n=sizeof(X)/sizeof(X[0]);

cout<<count(X,n,k);

return 0;

}