**Assignment-50: A Job Ready Bootcamp in c++,DSA and IOT**

**unordered\_map**

1. Write a c++ program to demonstrate functionality of unordered\_map.

Sol – 1.

#include<bits/stdc++.h>

using namespace std;

int main()

{

unordered\_map<string,double>umap={{"One",1},{"Two",2},{"Three",3}};

umap["PI"]=3.14;

umap.insert(make\_pair("e",2.718));

string s="PI";

if(umap.find(s)==umap.end())

cout<<s<<" not found!"<<endl;

else

cout<<"Found "<<s<<endl;

s="lambda";

if(umap.find(s)==umap.end())

cout<<s<<" not found!"<<endl;

else

cout<<"Found "<<s<<endl;

unordered\_map<string,double>::iterator it;

cout<<"All Elements : "<<endl;

for(it=umap.begin();it!=umap.end();it++)

{

cout<<it->first<<" "<<it->second<<endl;

}

return 0;

}

2. Given a string, Find the 1st repeated word in a string using unordered\_map.

Example:

Input : "Ravi had been saying that he had been there"

Output : had

Input : "Ravi had been saying that"

Output : No Repetition

Sol – 2.

#include<bits/stdc++.h>

using namespace std;

void solve(string s)

{

unordered\_map<string,int>mp;

string t="",ans="";

for(int i=s.length()-1;i>=0;i--)

{

if(s[i]!=' ')

{

t+=s[i];

}

else

{

mp[t]++;

if(mp[t]>1)

ans=t;

t="";

}

}

mp[t]++;

if(mp[t]>1)

ans=t;

if(ans!="")

{

reverse(ans.begin(),ans.end());

cout<<ans<<"\n";

}

else

{

cout<<"No repetition"<<endl;

}

}

int main()

{

string s="Ravi had been saying that he had been there";

solve(s);

return 0;

}

3. Write a c++ program to find freq of every word using unordered\_map.

Sol – 3.

#include<bits/stdc++.h>

using namespace std;

void printfreq(string s)

{

unordered\_map<char,int>mp;

for(int i=0;s[i];i++)

{

if(mp.find(s[i])==mp.end())

{

mp.insert(make\_pair(s[i],1));

}

else

{

mp[s[i]]++;

}

}

cout<<"Character\tFrequency"<<endl;

for(auto& it:mp)

{

cout<<it.first<<"\t\t"<<it.second<<endl;

}

}

int main()

{

string s="Gurudev Singla";

printfreq(s);

return 0;

}

4. Write a c++ program to demonstrate implementation of find function in

unordered\_map.

Sol – 4.

Above que have implementation of find function

5. Given a positive integer N, the task is to print the nearest power of 2 of the

frequencies of each digit present in N. If there exists two nearest powers of 2 for any

frequency, print the larger one using unordered\_map.

Example:

Input: N = 344422

Output:

2 -> 2

3 -> 1

4 -> 4

Explanation:

Frequency of the digit 3 is 1. Nearest power of 2 is 1.

Frequency of the digit 4 is 3. Nearest power of 2 is 4.

Frequency of the digit 2 is 2. Nearest power of 2 is 2.

Sol – 5.

#include<bits/stdc++.h>

using namespace std;

void nearestPowerOfTwo(string &s)

{

int n=s.size();

unordered\_map<char,int>mp;

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

{

mp[s[i]]++;

}

for(auto &it : mp)

{

cout<<it.first<<"-->";

int lg=log2(it.second);

int a=pow(2,lg);

int b=pow(2,lg+1);

if((it.second-a)<(b-it.second))

{

cout<<a<<endl;

}

else

{

cout<<b<<endl;

}

}

}

int main()

{

string s="344422";

nearestPowerOfTwo(s);

return 0;

}

6. Given two integers L, R, and an integer K, the task is to print all the pairs of Prime

Numbers from the given range whose difference is K using unordered\_map.

Example:

Input: L = 1, R = 19, K = 6

Output: (5, 11) (7, 13) (11, 17) (13, 19)

Explanation: The pairs of prime numbers with difference 6 are (5, 11), (7, 13), (11,

17), and (13, 19).

Sol – 6.

#include<bits/stdc++.h>

using namespace std;

void findPrimeNos(int l,int r,unordered\_map<int,int>&m)

{

for(int i=l;i<=r;i++)

{

m[i]++;

}

if(m.find(1)!=m.end())

{

m.erase(1);

}

for(int i=2;i<=sqrt(r);i++)

{

int mult=2;

while((i\*mult)<=r)

{

if(m.find(i\*mult)!=m.end())

{

m.erase(i\*mult);

}

mult++;

}

}

}

void getPrimePairs(int l,int r,int k)

{

unordered\_map<int,int>mp;

findPrimeNos(l,r,mp);

for(auto &it:mp)

{

if(mp.find(it.first+k)!=mp.end())

{

cout<<"("<<it.first<<","<<it.first+k<<")";

}

}

}

int main()

{

int l=1,r=19;

int k=6;

getPrimePairs(l,r,k);

return 0;

}

7. Create an unordered\_map and Initialize it from another map using the copy

Constructor

Sol – 7.

#include<bits/stdc++.h>

using namespace std;

int main()

{

unordered\_map<string,string>m1;

m1["Ground"]="Grass";

m1["Floor"]="Cement";

m1["Table"]="Wood";

unordered\_map<string,string>m2(m1);

for(auto x:m2)

{

cout<<x.first<<"-->"<<x.second<<endl;

}

return 0;

}

8. Create an unordered\_map and Initialize it using assignment and subscript operator

Sol – 8.

Done in above que

9. Given string str, the task is to find the minimum count of characters that need to be

deleted from the string such that the frequency of each character of the string is

unique using unordered\_map.

Example:

Input: str = “ceabaacb”

Output: 2

Explanation:

The frequencies of each distinct character are as follows:

c —> 2

e —> 1

a —> 3

b —> 2

Possible ways to make frequency of each character unique by minimum number of

moves are:

● Removing both occurrences of ‘c’ modifies str to “eabaab”

● Removing an occurrence of ‘c’ and ‘e’ modifies str to “abaacb”

Therefore, the minimum removals required is 2.

Sol – 9.

#include<bits/stdc++.h>

using namespace std;

int minCharDel(string &s,int n)

{

unordered\_map<char,int>mp;

priority\_queue<int>pq;

int cntChar=0;

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

{

mp[s[i]]++;

}

for(auto it:mp)

{

pq.push(it.second);

}

while(!pq.empty())

{

int f=pq.top();

pq.pop();

if(pq.empty())

{

return cntChar;

}

if(f==pq.top())

{

if(f>1)

{

pq.push(f-1);

}

cntChar++;

}

}

return cntChar;

}

int main()

{

string s="abbbcccd";

int n=s.length();

cout<<minCharDel(s,n);

return 0;

}

10. Given an array arr[] consisting of N integers, the task is to find the maximum element

with the minimum frequency using unordered\_map.

Example:

Input: arr[] = {2, 2, 5, 50, 1}

Output: 50

Explanation:

The element with minimum frequency is {1, 5, 50}. The maximum element among

these element is 50.

Sol – 10.

#include<bits/stdc++.h>

using namespace std;

int maxElementWithMinFreq(int a[],int n)

{

unordered\_map<int,int>mp;

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

{

mp[a[i]]++;

}

int minFreq=INT\_MAX;

int maxValue=INT\_MIN;

for(auto x:mp)

{

int num=x.first;

int freq=x.second;

if(freq<minFreq)

{

minFreq=freq;

maxValue=num;

}

else if(freq==minFreq&&maxValue<num)

{

maxValue=num;

}

}

return maxValue;

}

int main()

{

int arr[]={2,2,50,5,1};

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

cout<<maxElementWithMinFreq(arr,n);

return 0;

}