**Assignment-52: A Job Ready Bootcamp in c++,DSA and IOT**

**unordered\_multimap**

1. Write a C++ program to initialise the unordered\_map and print it on the screen.

Sol – 1.

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

int main()

{

unordered\_multimap<int,string>ump;

ump.insert({1,"hello"});

ump.insert({2,"hey"});

ump.insert({1,"hie"});

for(auto x:ump)

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

return 0;

}

2. Write a C++ program to copy the data of one unordered\_multimap into another and print it

on the screen.

Sol – 2.

#include<iostream>

#include<unordered\_map>

using namespace std;

int main()

{

unordered\_multimap<int,string>ump,ump1;

ump.insert({1,"hello"});

ump.insert({2,"hey"});

ump.insert({1,"hie"});

ump1=ump;

for(auto x:ump1)

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

return 0;

}

3. Write a C++ program to illustrate the reverse function in unordered\_multimap.

Sol – 3.

#include<iostream>

#include<unordered\_map>

using namespace std;

int main()

{

unordered\_multimap<int,string>ump;

ump.insert({1,"hello"});

ump.insert({2,"A"});

ump.insert({1,"hie"});

for(auto x:ump)

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

cout<<"After reserve :"<<endl;

ump.reserve(3);

for(auto x:ump)

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

return 0;

}

4. Given a string of words, the task is to find the frequencies of the individual words

using unordered\_map.

Example:

Input: str = “Ineuron for Ineuron Ineuron quiz practice qa for”;

Output: Frequencies of individual words are

(qa, 1)

(quiz, 1)

(practice, 1)

(Ineuron, 3)

(for, 2)

Sol – 4.

#include<iostream>

#include<map>

#include<string>

using namespace std;

int main()

{

string s="My name is Gurudev Singla My";

string word="";

map<string,int>mp;

for(int i=0;i<s.size();i++)

{

if(s[i]==' ')

{

if(mp.find(word)==mp.end())

{

mp.insert(make\_pair(word,1));

word="";

}

else{

mp[word]++;

word="";

}

}

else

word+=s[i];

}

if(mp.find(word)==mp.end())

{

mp.insert(make\_pair(word,1));

word="";

}

else

{

mp[word]++;

word="";

}

for(auto x:mp)

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

return 0;

}

5. Given an array arr[] of size N and three integers X, Y and K, the task is to count the

number of pairs (i, j) where i < j such that (arr[i] \* X + arr[j] \* Y) = K complete the task

using unordered\_map.

Example:

Input: arr[] = {3, 1, 2, 3}, X = 4, Y = 2, K = 14

Output: 2

Explanation: The possible pairs are: (1, 2), (3, 4).

For i = 1, j = 2, Value of the expression = 4 \* 3 + 2 \* 1 = 14.

For i = 3, j = 4, Value of the expression = 4 \* 2 + 2 \* 3 = 14.

Input: arr[] = [1, 3, 2], X = 1, Y = 3, K = 7

Output: 1

Explanation: The possible pairs are: (1, 2).

For i = 1, j = 2, Value of the expression = 1 \* 1 + 2 \* 3 = 7.

Sol – 5.

#include<iostream>

using namespace std;

void morethanNbyK(int arr[],int n,int k,int x,int y)

{

int count=0;

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

{

for(int j=i+1;j<n;j++)

{

int first=arr[i]\*x;

int second=arr[j]\*y;

int sum=first+second;

if(sum==k)

count++;

}

}

cout<<endl<<"Count = "<<count<<endl;

}

int main()

{

int arr[]={3,1,2,3};

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

int k=14;

int x=4,y=2;

morethanNbyK(arr,n,k,x,y);

return 0;

}

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

with the minimum frequency using unordered\_map.

Sol – 6.

Already Done!!

7. Given an array arr[] consisting of N positive integers, the task is to find the count of

unique pairs (i, j) such that the sum of arr[i] and the reverse(arr[j]) is the same as the

sum of reverse(arr[i]) and arr[j] using unordered\_map.

Example:

Input: arr[] = {2, 15, 11, 7}

Output: 3

Explanation:

The pairs are (0, 2), (0, 3) and (2, 3).

● (0, 2): arr[0] + reverse(arr[2]) (= 2 + 11 = 13) and reverse(arr[0]) + arr[2](= 2 +

11 = 13).

● (0, 3): arr[0] + reverse(arr[3]) (= 2 + 7 = 9) and reverse(arr[0]) + arr[3](= 2 + 7

= 9).

● (2, 3): arr[2] + reverse(arr[3]) (= 11 + 7 = 18) and reverse(arr[2]) + arr[3](= 11

+ 7 = 18).

Input: A[] = {22, 115, 7, 313, 17, 23, 22}

Output: 6

Sol – 7.

#include<iostream>

#include<unordered\_map>

using namespace std;

int reverse(int n)

{

int rev=0;

while(n)

{

rev=rev\*10+n%10;

n/=10;

}

return rev;

}

void countPairs(int A[],int N)

{

unordered\_map<int,int>ump;

int pairs=0;

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

{

int val=A[i]-reverse(A[i]);

pairs+=ump[val];

ump[val]++;

}

cout<<pairs;

}

int main()

{

int arr[]={22, 115, 7, 313, 17, 23, 22};

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

countPairs(arr,n);

return 0;

}

8. Given an array A[] consisting of positive integers, the task is to find the only array

element with a single occurrence using unordered\_map.

Sol – 8.

Already done!!

9. Given an array arr[], the task is to find the count of array elements whose squares

are already present in the array using unordered\_map.

Sol – 9.

#include<iostream>

#include<unordered\_map>

using namespace std;

void countSquares(int arr[],int n)

{

int count=0;

unordered\_map<int,int>m;

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

{

m[arr[i]]++;

}

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

{

int square=arr[i]\*arr[i];

if(m[square]!=0)

count+=1;

}

cout<<count;

}

int main()

{

int arr[]={2,4,5,20,16};

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

countSquares(arr,n);

return 0;

}

10. 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 – 10.

Already Done!!