**C++ Notes**

Int: 10^9

Long ing: 10^12

Long long int: 10^18

Char – it get only single char

fixed: used to fix the decimal or float value in c++

setprecision(0): used to set 0 precision in c++

1e5 : it is equal 10 power 5

similar eg: 1e4 etc

**String**:

getline(cin,str); -can be used to get string from user

cin.ignore(); - used to ignore the space

strreverse.push\_back(str2); -can be used to push char in a string

**Array**:

* local array can have size upto 10^5 or 1e5
* global array can have upto 1e7 using constant variable for size

eg :

const int size=1e7;

int array[n];

int main(){…………}

**In array , if we call array variable without index then it display address of the first index of first byte of that array**

Eg:int a[15];

Cout<<a;🡪displaying address of the array

**call by value**: sending value to the function

**call by reference** : sending address of the value to the function

->Array always pass by reference to the function

**Function**: swap(),min(),max()

**Note:**

int a;🡪4byte

&a=&a+1 🡪 it add 4byte to address a, so if address value of a is 1 then it add 4 byte to it and become 5 so now address of a is 5

**Pointer**: it is used to store address of a variable

Eg:

int \*p; int x=4;

P=&x;🡪storing address of the variable in p

Cout<<\*p;🡪displaying value of p address or displaying value of x

**Double pointer**: it is used to store the address of a pointer variable

Eg: int \*p,\*\*pp;

Int x=4;

p=&x;

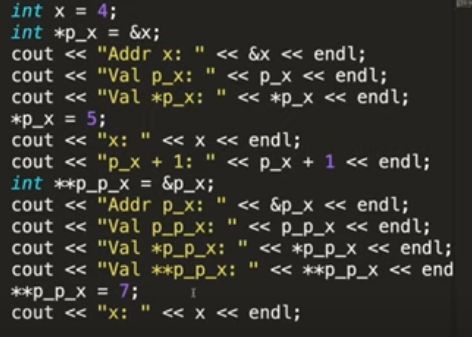
pp=&p;

Cout<<pp;🡪displaying address of p value;

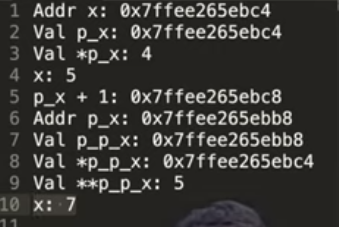
Cout<<\*pp;🡪displaying address of x value;

Cout<<\*\*pp;🡪displaying value of x

Code:



Output:



Ep 11 Why Print answer modulo 10^9+7(Module Arithmatic)

Ep 12 Pre-Comutation Techniques basics and Hashing

**Hashing**: it can be used for reducing time complexity using hash array index.

For hashing index should start from 1

Ep 13 Pre-Computation using Prefix Sum in 1D/2D Arrays

Ep 14 Problem gcd

**Note:** Prefixsum array used to reduce the time complexity of the problem

**Note:** In C++, you can find gcd using predefined function \_\_gcd(a,b) which is O(logn).

Ep 15 Problem

Ep 16 Prefix Sum + Hashing HARD Question using brute force method palindrome

**Mergesort:** In this he is using different method to merge two array using int\_max value at both array ends.

**Ep 20**

**🡪 C++ STL** or **Standard Template Library** is a very important and crucial thing to learn if you want become good at competitive Programming. In this video i have given a full over view of things present in C++ STL and in upcoming videos we will study STL in detail.

* Containers
* Iterators
* Algorithms
* Fuctions

**Containers**

**Sequential Orderlist Unorder**

* Vectors Maps unorderMet
* Stack Multimap unorderSet
* Queue Set
* Pair(Not a container) Multiset

Ep 21

**Pair:** It is used to store pair value

Ex: pair<int,string> p;

To make pair we need to use **make\_pair** function

Eg: p=make\_pair(2,”abc”);

Or p={2,”abc”};

To print first pair use **variable.first** snytex and for second pair use variable.second

Eg: cout<<p.first<<””<<p.second;

**We use pair to maintain the relation ex; two array**

**To use pair as array:**

Pair<int,int> p\_arrray[n];

P\_array[0]={3,2};

**Vector:** It has dynamic sizeand same space limit as array

Eg:

vector<int> v;

**Vector Funciton**

v.**push\_back**(x); //O(1) it is used to add value in the vector

v.**pop\_back**(x); //it is used to remove value at last

v.**size**(); // it is used to find the size of the vector

Note: In function pass reference so that it will not create a copy of the vector

Eg:

Void refunction(vector<int> &v1)//if you are not using reference then it will create copy of { the vector which will be O(n) process

//print the vector using for loop

}

Int main(){

vector<int> v;

Refunction(v)

}

**To declare specific size vector**:

vector<int> v(10); // it will assign value as zero.

Output: 0 0 0 0 0 0 0 0 0 0

To add specific value in vector will declaration of the Specific size:

Eg:

vector<int> v(10,3); // it will assign value as three.

v.push\_back(7);

Output: 3 3 3 3 3 3 3 3 3 3 7

**To create copy of vector array:**

Vector<int> v1= v; // **O(n) To create a copy of vector**

**Ep 22**

**Nesting in a vector:**

Eg:

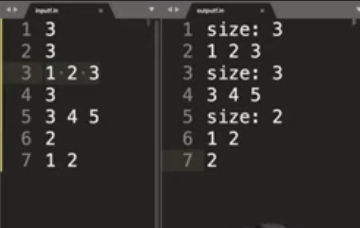
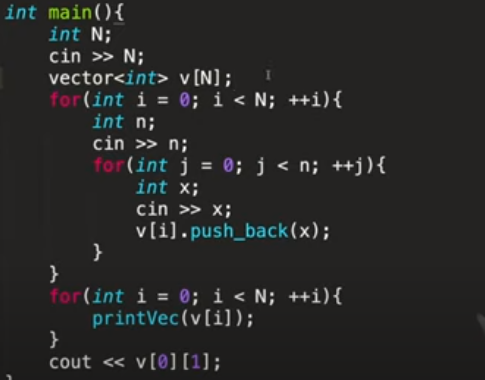
vector <pair<int,int>> v={ {1,2},{3,4},{5,6}};

cout<<v[0].first<<” “<<v[0].second; // output:1 2

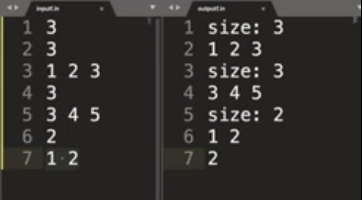
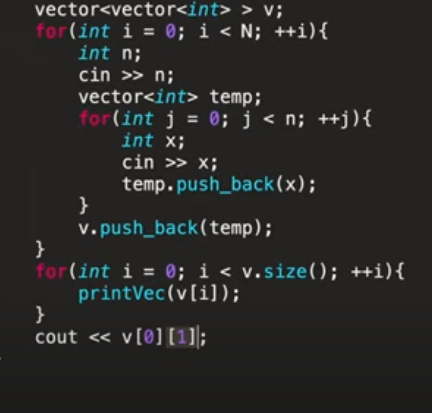
or

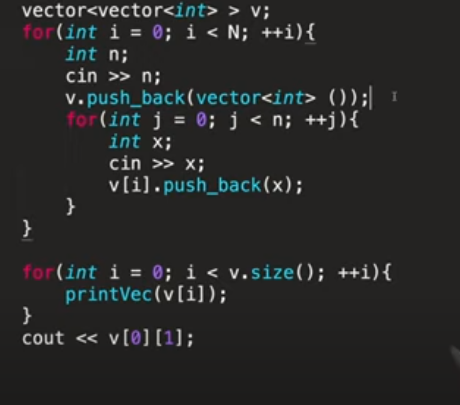
v.**push\_back**( {7,8} ); or v.push\_back(**make\_pair**(7,8));

**Array of vectors:** Vector<int> v[10];



**Vector of vector:**



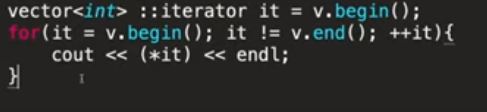


**Iterators:**

**Declaration of iterators:**

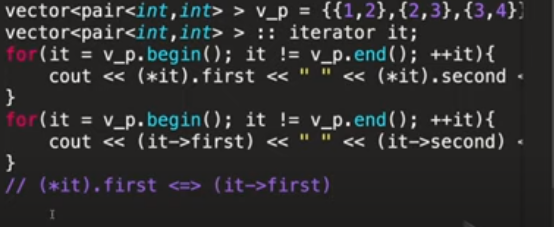
Vector<int> : : iterator it = v.begin(); // v.begin() represents v[0] and v.end(); represents

v [v.size()-1];



**To print iterators use : \* operator**

**Note:** In iterators it++ and it+1 is different



**Auto and Range based loops: //**Need c++ version 11 to run this loops

Range based loops:

Eg:

for( int &value: v){ // **it doesn’t create a copy of vector**

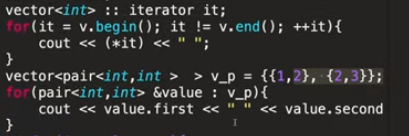
value++;

}

for( int value: v){ // **it create a copy of vector**

Cout<<value ;

}



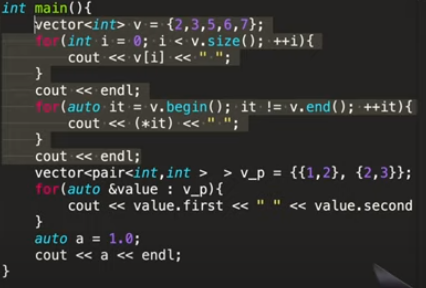
**Auto :** This keyword take automatic value for variable and automatically consider what types of value is this like vector, pair, iterator etc.

It reduces the size of the code

Eg:

auto a;

Cout<<a; // o/p: 1

**Ep 25**

**Maps:** It store key , value pair and stores the value in sorted order. It has unique key.

🡪 it uses tree

**Function**:

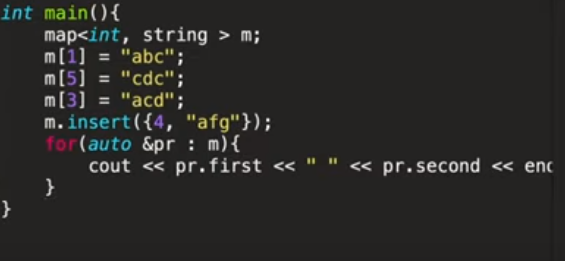
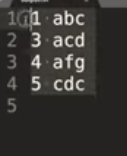
variable\_name.**insert**({1,”abc”});

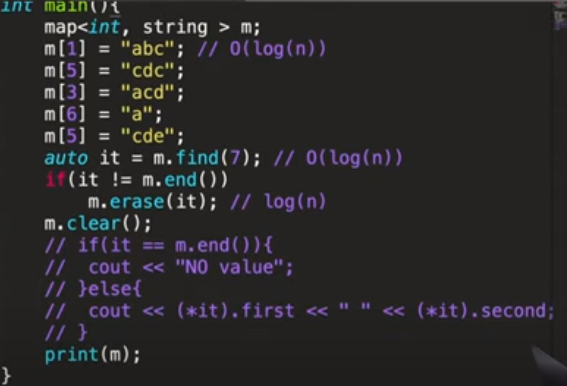
m[1]=”abc”; // **O(log n)** complexity to **store and display or access** a value

variable\_name.**find**(1); // **O(log n) this fun return iterator**

variable\_name.**erase**(1); // **O(log n)**

variable\_name.**clear()**;



**Note**:

map<string,string> m;

m[“abcd”]=”abc”; //it take s.size()\*log(n ) complexity to store value based on string length

**EP 26**

**Unordered Maps:** It store key , value pair and stores the value in unsorted order. It has unique key.

🡪It uses hash table

**Function**:

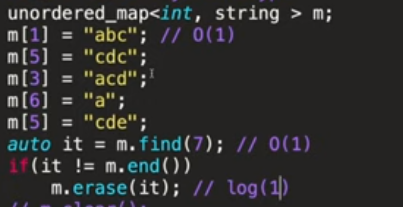
variable\_name.**insert**({1,”abc”});// **O(1)**

**Difference b/w unordered map vs maps**

**🡪** Time Complexity

**🡪**Inbuild Implemenation

**🡪** Valid Keys Datatype



**Note: 3 different valid datatype defined as below**

unordered\_map< pair<int , int> ,string > m;

// it **throws error** due to it cannot find hash value to pair , vector etc but it can find hash value of int,string,double,char….

map<pair< int, int >,string > m;

// it doesn’t **throws error** due to it is not find the hash value to pair , vector etc but actually it comparing any type of datatype

**MultiMap:** it is similar to maps but different is it is not unique.

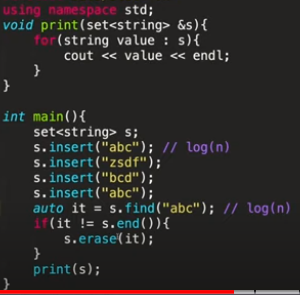
EP 27

**SETS:** It store unique value and sort the value automatic.

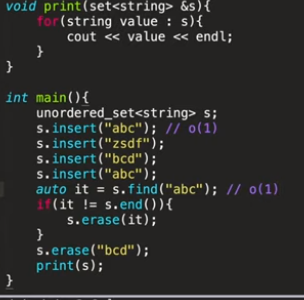
set<string> s;

s.insert(“abc”);//log(n)

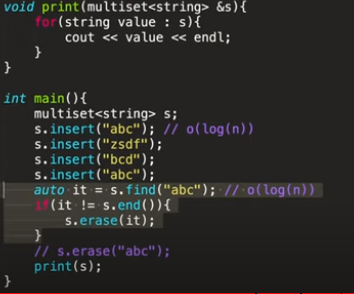
auto it=s.find(“abc”);



**Unordered set:**



**Multiset:**

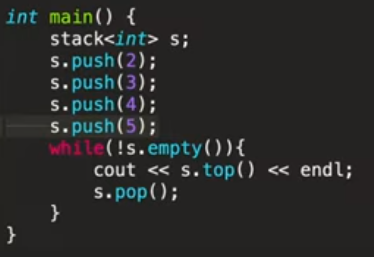
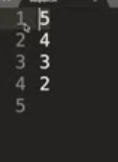


**// this will delete only one value**

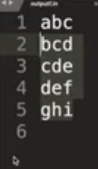
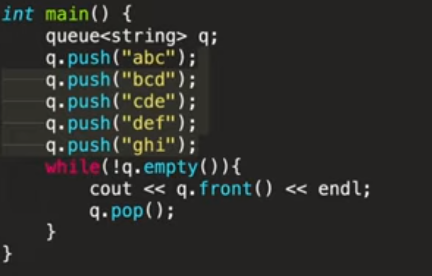
**//this will delete all the value which is equal to “abc”**

**Ep 30**

**Stack:**

**Queue:**



**Inbulid Sort:** It is also called **Introsort.** Syntax: sort()

**O(log n) worst complicity**

🡪 It is combination quick sort, heap sort, inseration sort. When array has less n value it use inseration sort and when array has large n value it use heap sort else quick sort for medium n array.

Eg: int array[10];

sort(a , a+n); // ’a’ represent a[0] pointer mean address and a+n represent last array value.

/// in vector

vector <int> a(n);

sort(a.begin()+1 , a.end()); // a.begin()+1 represent a[1] vector

**Comparator Operator:**

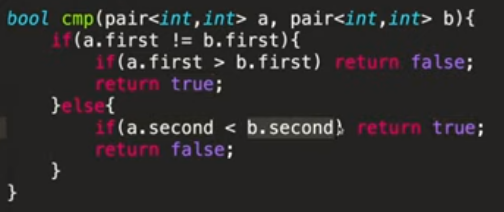
We can use comparator operator in sorting and other area.

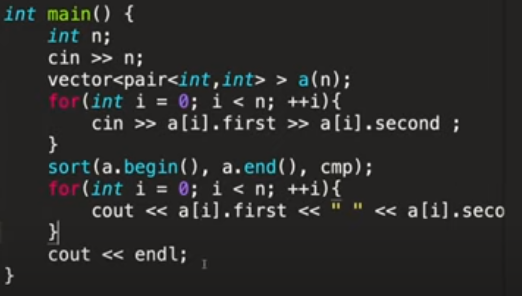
**eg: sort(a.begin(),a.end(),cmp);**

In sorting function it takes only **false as return type** which is consider as true value

We have inbuild function for this for which we can refer any website

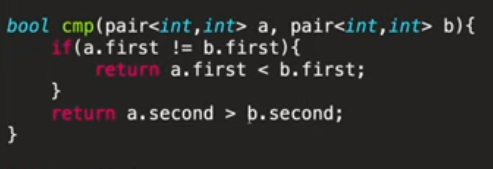
See this example below,





**Or you can use another method**

**This will return same as above method**



**EP 35**

**UPPER BOND AND LOWER BOND:** It needed **sorted array** to perform this operation.

In lower bond, if you find a X value present in the array then if it is **present then it will show X** value **else it will show value greather than X**.

In upper bond, if you find a X value present in the array then it will **show greather value than x**

**0(log n) 🡪 time complexity**

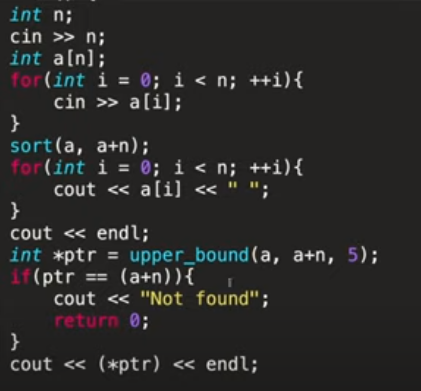
**For array example:**

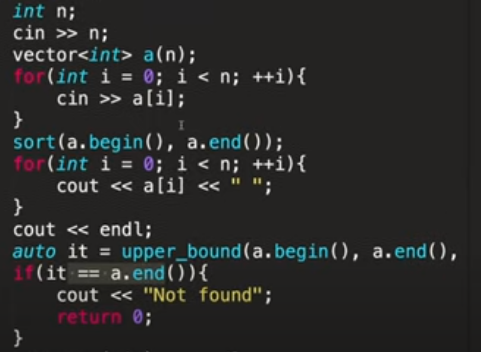
**Upper bond:**

**Array 4 5 5 7 8 25**

**X=5 then o/p 7**

**X=6 then o/p 7**

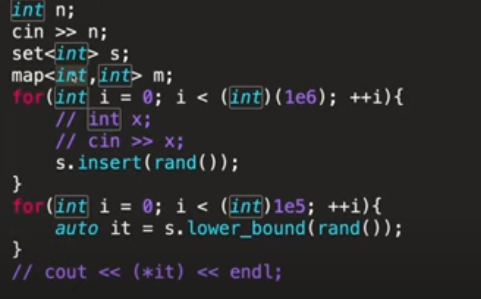


**For vector**

**//auto it=upper\_bond(a.begin(),a.end(),5);**

**For Set and Map:**

**Syntax: variable\_name . lower\_bond(value);**



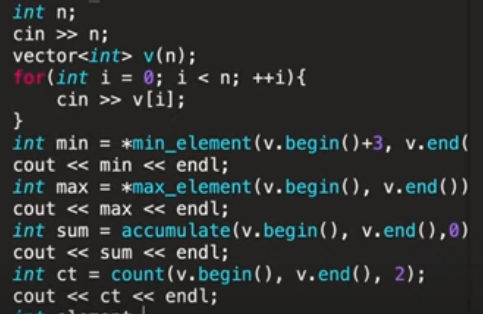
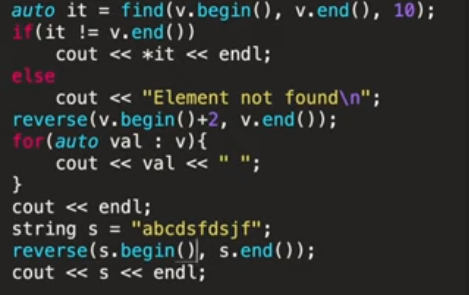
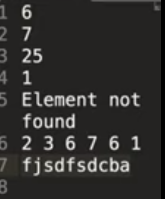
**In map, bound is set to the key it will check the key to find the value.**

**EP 36**

**Inbuild Function for vector and array:**

* + - **min\_element()** find minimum element in array or vector **, it return pointers**
    - **max\_element()** find maximum element in array or vector**, it return pointers**
    - **accumulate()** find the sum of array or vector**,**
    - **count()** find the count of single element present in the array**,**
    - **reverse()** – To reverse the string and array
    - **find()** find the value in the array present or not, **it return pointers**

**It is O(n) time complexity**

EP 37

**Labda Funciton:**

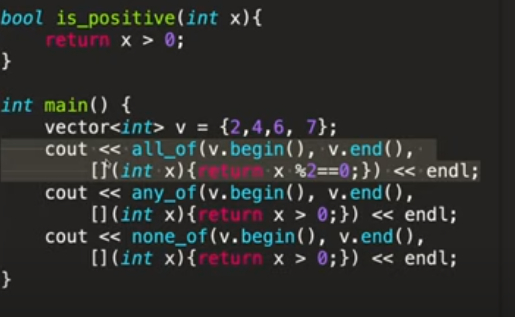
Syntax: [](datatype){function}(variable passed);

Eg:

* cout<<[](int x){return x+2}(2); // O/P 4
* cout<<[](int x,int y){return x+y}(2,7); // O/P 9
* auto sum=<<[](int x,int y){return x+y};

cout<<sum(2,3); //O/P 5

**Allof and Anyof , Noneof:**



Cout<<allof(v.begin(),v.end(), is\_positive);

O/P: 1

**EP 38** Generate Parenthesis: Advanced Recursion and Backtracking \*\*\*

**EP 39** Subset Generation using Recursion and backtracking \*\*\*

**EP 40** Binary Search & How I write it

🡪 Binary search needed Monotonic function

O(log N)

Monotonic function: Function which maintain the given order.

Eg : sorted array or order array

1,3,6,8,10 is MF or 1,2,3,4,5

**EP 41** Implement Upper Bound & Lower Bound with Binary Search

**EP 42** Nth Root of a Number using Binary Search \*\*\*

**EP 43** Advanced Binary Search with Predicate Function \*\*\* - CutTree

🡪Predicate Function: The function return True/False

**EP 44** Advanced Binary Search with Predicate Function - SPOJ AGGRCOW \*\*\*

**EP 45** Explanation How to Calculate Time Complexity In Recursion ?

**EP 47**

# Binary Numbers and Bits Basics

# Left shift <<

# 3🡪11

# 3<<1 🡪 110

# 3<<2 🡪 1100

# Right shift >>

# 6🡪111

# 6>>1 🡪 11

# 6>>2 🡪 1

# tidal ~

# Int a=5🡪101 // reverse the bit

# ~a // 010

# We can take value of 2 Power n using 1<<n

# Eg: 2 Power 2 =4

# 1<<2 = 1000 bits 🡪 4

# Toggle operator: ^ invert the bit 1🡪 0

# 

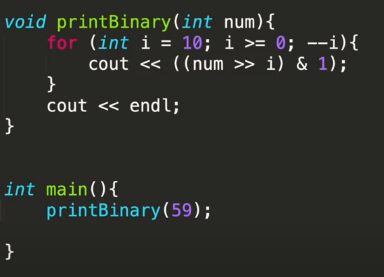
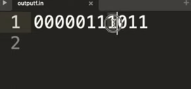
Int 🡪 signed int : 1 bit is reserved for sign eg: +/- \*\*\*

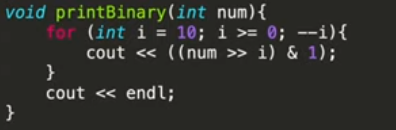
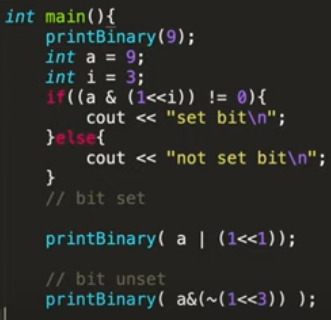
Unsigned int : which will not take any bit for sign

**EP 48**

1 🡪 set bit

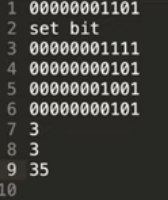
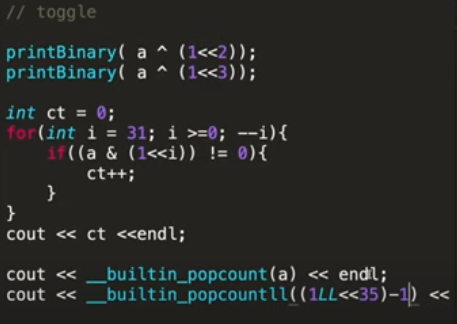
0 🡪 unset bit



\_\_builtin\_popcount() – Function used to count set for int

\_\_builtin\_popcountll() – Function used to count set for long



**EP 49** 6 Amazing BIT Manipulation Ticks You must Know

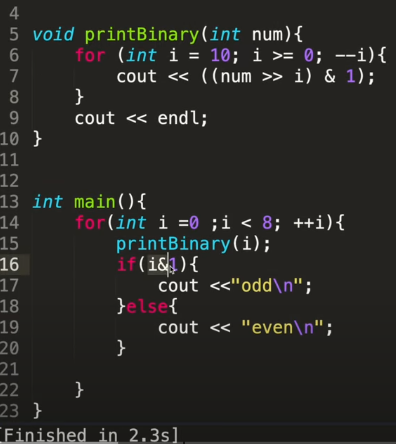
1. Odd or Even: Using & operator we can find a number is odd or even,without using % operator

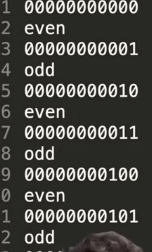
Eg: 1 = 0000 0001 bits

2= 0000 0010 bits

For odd number 0th bit will be 1 always so using AND & operation with 1 we can get whether given no is odd or even

2&1 = 0 \*\*\*





1. Multiple / Divide by 2:

we can use right shift for Division and left shift for multiplication.

Eg: 5/2=2 and 5\*10=10

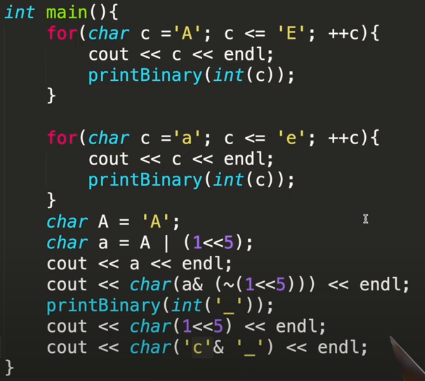
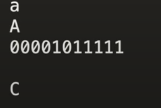
5>>1 =2\*\*\*

5<<1=10\*\*\*

1. Upper Case Lower Case Conversion :

Eg: For A = 0000 1000 001 a= 0000 1100 001

So highlighted bit space is different from each other so we can convert A to a using | OR operation with 1<<5 (0000 0100 000) and visa vera for a to A & AND operation with ~(1<<5)

 O/P

Another ways to convert a to A is

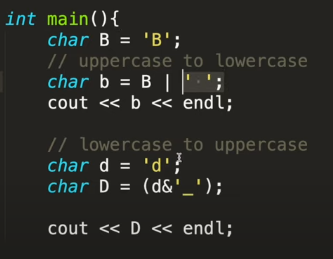
Eg : using ‘\_’ underscore with & operator

Char(‘c’ &’\_’); // O/P: C

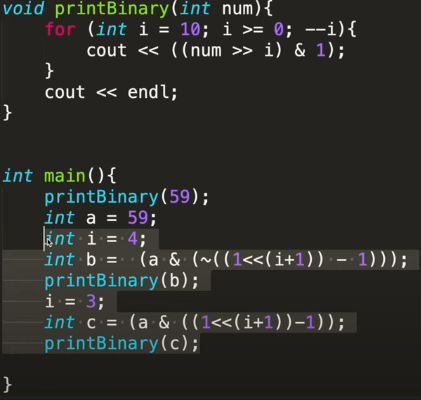
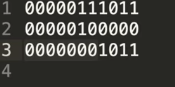
convert A to a is

Using ‘ ‘ Space with |operator

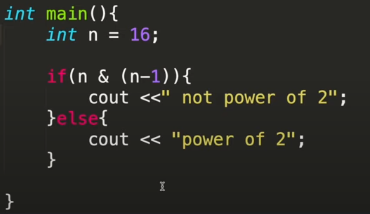
Char(‘C’ &’ ’); // O/P: a

1. Clear LSB and Clear MSB:

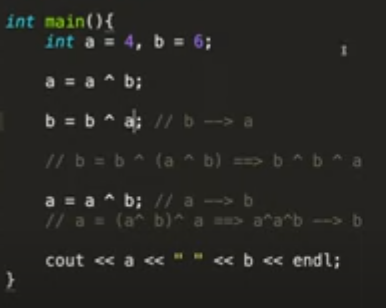
 

1. Check power of 2:

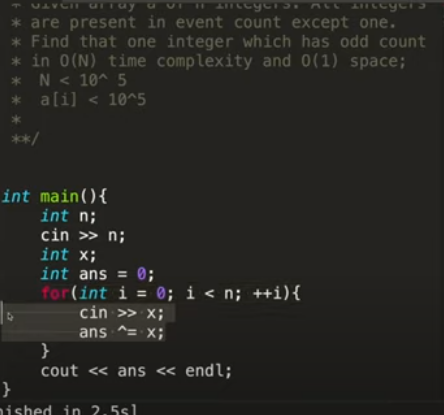
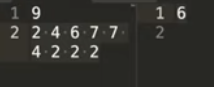
 O/P: 

**EP 50** Power of XOR(^): 0 ^ 1 🡪 1 1 ^ 1🡪0

Swapping of two variable number:

 **O/P: 6 4**

**Problem: Given array has n interger where all no has pair expect one. find the odd one out.**



**EP 51** What is Bit Masking with example Question

EP 54.1 Binary Exponentiation : Recursive Method

Pow(x,Y)- c++ Method return double, sometime it don’t return proper percision so using this method for CP is not good.

So we can solve this using Binary Exponentiation.

Using Divide and conquer method:

Eg: 216 for odd power 313 for odd power

216 🡪28 \*28 313 🡪3\*312

28 🡪24 \*24 312 🡪36 \*36

24 🡪22 \*22 36 🡪33 \*33

22 🡪21 \*21  33 🡪3\*32

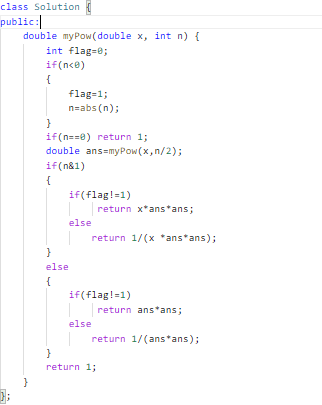
32🡪31 \*31

If power is negative then

Eg : 2-2 🡪 21/2  so n=1/2 now, or

We can convert power into Positive value and return the value by 1/22

if **power is negative then**: Convert n into positive number using abs() method and answer should return 1/myPow(x,n);



EP 54.2 Binary Exponentiation : Iterative Method

EP 57 Basics of Factors and Divisors related problems

EP 58 Prime Check & Prime Factorisation

EP 78 Breadth First Search

Code:

#include <bits/stdc++.h>

using namespace std;

const int N =1e5+10;

vector<int> g[N];

int vis[N];

int level[N];

void bfs(int source){

queue<int> q;

q.push(source);

vis[source]=1;

while(!q.empty()){

int cur\_v=q.front();

q.pop();

cout<<cur\_v<<" ";

for(int child:g[cur\_v]){

if(!vis[child]){

q.push(child);

vis[child]=1;

level[child]=level[cur\_v]+1;

}

}

}

}

int main(){

int n;

cin>>n;

for(int i=0;i<n-1;++i){

int x,y;

cin>>x>>y;

g[x].push\_back(y);

g[y].push\_back(x);

}

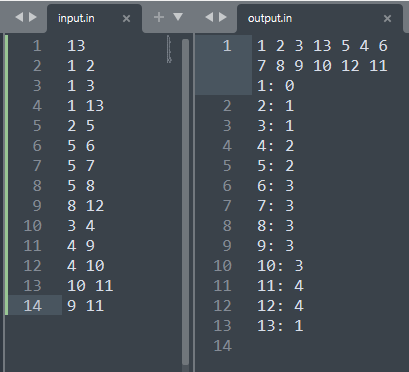
bfs(1);

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

cout<<i<<": "<<level[i]<<endl;

}

}



EP 79 How to find shorest path using Breadth First Search

EP 80 0-1 BFS: What is it, How to Code with Example Question

EP 83 Floyd Warshall Algorithm: All Pair Shortest Path

#include <bits/stdc++.h>

using namespace std;

const int N =510;

const int INF=1e9+10;

int dist[N][N];

int main()

{

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

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

if(i==j) dist[i][j]=0;

else dist[i][j]=INF;

}

}

int n,m;

cin>>n>>m;

//To GET Input

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

int x,y,wt;

cin>>x>>y>>wt;

dist[x][y]=wt;

}

// Logic

for(int k=1;k<=n;++k){

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

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

if(dist[i][k]!=INF && dist[k][j]!=INF)

dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);

}

}

}

// To Print

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

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

if(dist[i][j]==INF)

cout<<"I"<<" ";

else

cout<<dist[i][j]<<" ";

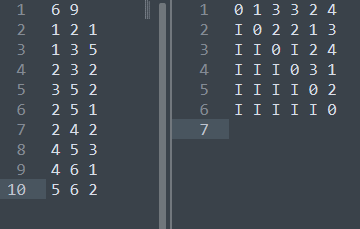
}

cout<<endl;

}

}

OUTPUT:



EP 86 Disjoint set Union:

#include <bits/stdc++.h>

using namespace std;

const int N =1e5+10;

int size[N];

int parent[N];

void make(int v){

parent[v] = v;

size[v] = 1;

}

int find(int v){

if(v == parent[v]) return v;

// Path Compression

return parent[v]=find(parent[v]);

}

void Union(int a,int b){

a = find(a);

b = find(b);

if(a != b){

// Union by size

if(size[a] < size[b])

swap(a , b);

parent[b]=a;

size[a] += size[b];

}

}

int main(){

int n,k;

cin>>n>>k;

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

make(i);

}

while(k--){

int u,v;

cin>>u>>v;

Union(u,v);

}

int ct=0;

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

if(find(i)==i) ct++;

}

cout<<ct<<endl;

}

Output:

2

Input:

4

2

1 2

4 1

EP 87 Kruskal's Algorithm: Minimum Spanning Tree

#include <bits/stdc++.h>

using namespace std;

const int N =1e5+10;

int size[N];

int parent[N];

void make(int v){

parent[v] = v;

size[v] = 1;

}

int find(int v){

if(v == parent[v]) return parent[v];

// Path Compression

return parent[v]=find(parent[v]);

}

void Union(int a,int b){

a = find(a);

b = find(b);

if(a != b){

// Union by size

if(size[a] < size[b])

swap(a , b);

parent[b]=a;

size[a] += size[b];

}

}

int main(){

int n,m;

cin>>n>>m;

vector<pair<int,pair<int,int>>> edges;

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

int u,v,wt;

cin>>u>>v>>wt;

edges.push\_back({wt,{u,v}});

}

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

sort(edges.begin(),edges.end());

int total\_cost=0;

for(auto &edge:edges){

int wt=edge.first;

int u=edge.second.first;

int v=edge.second.second;

if(find(u) ==find(v)) continue;

Union(u,v);

total\_cost+=wt;

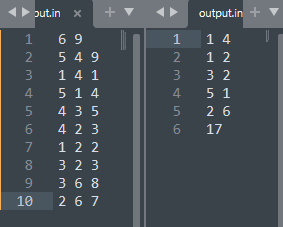
cout<<u<<" "<<v<<endl;

}

cout<<total\_cost;return 0;

}

Output:



EP 88 Google Interview Hard Question | Minimum Spanning Tree

EP 95 0-1 Knapsack Tutorial With Code: Dynamic Programming

**D - Knapsack 1**

#include <bits/stdc++.h>

using namespace std;

int wt[105];

int val[105];

long long dp[105][100005];

long long funt(int ind,int wt\_left){

if(wt\_left==0) return 0;

if(ind<0) return 0;

if(dp[ind][wt\_left] != -1) return dp[ind][wt\_left];

long long ans;

// Dont chooose item at ind

ans=funt(ind-1,wt\_left);

// chooose item at ind

if(wt\_left-wt[ind] >= 0)

ans=max(ans,funt(ind-1,wt\_left-wt[ind])+val[ind]);

return dp[ind][wt\_left]=ans;

}

int main(){

int n,w;

cin>>n>>w;

memset(dp,-1,sizeof(dp));

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

cin>>wt[i]>>val[i];

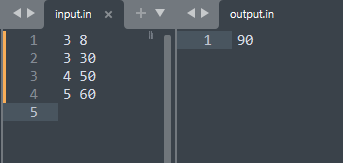
}

cout<<funt(n-1,w);

return 0;

}

**OUTPUT:**



EP:98 SUBSET SUM & PARTITION PROBLEM : Dynamic Programming

class Solution {

public:

    int dp[200][20000];

    bool func(int len,int sum,vector<int>& nums){

        if(sum==0) return true;

        if(len<0) return false;

        if(dp[len][sum] != -1) return dp[len][sum];

        bool ispossible=func(len-1,sum,nums);

        if(sum-nums[len]>=0)

            ispossible |= func(len-1,sum-nums[len],nums);

       return dp[len][sum]=ispossible;

    }

    bool canPartition(vector<int>& nums) {

        int sum=accumulate(nums.begin(),nums.end(),0);

        if(sum%2!=0) return false;

        sum/=2;

        memset(dp,-1,sizeof(dp));

        return func(nums.size()-1,sum,nums);

    }

};

Example 1:

Input: nums = [1,5,11,5]

Output: true

Explanation: The array can be partitioned as [1, 5, 5] and [11].

Example 2:

Input: nums = [1,2,3,5]

Output: false

Explanation: The array cannot be partitioned into equal sum subsets.