**1. My Template:**

#include <bits/stdc++.h>

using namespace std;

#define ll long long

#define pb push\_back

#define Sort(name) sort(name.begin(), name.end())

#define fast\_in\_out                   \

ios\_base::sync\_with\_stdio(false); \

cin.tie(0);                       \

cout.tie(0);

int main()

{

fast\_in\_out;

int t;

cin>>t;

for(int i=1;i<=t;i++)

{

}

}

**2. Reverse a linked List:**

while(nextNode!=NULL) // nextNode = head,prevNode = NULL, currentNode = head;

    {

        nextNode = nextNode -> next;

        currentNode->next = prevNode;

        prevNode = currentNode;

        currentNode = nextNode;

    }

    return prevNode;

};

**3. Deleting node without head:**

void deleteNode(ListNode\* node) {

    node->val = node->next->val;

    node->next= node->next->next;

}

2.Create Dynamic new LinkedList Node->

ListNode \*mergeNodes(ListNode \*head)

{

**ListNode \*ans;**

**ListNode \*ans\_head = ans;**

    int sum = 0;

    while (head != NULL)

    {

if (head->val != 0)

        {

            sum += head->val;

            head = head->next;

        }

        else

        {

            if (sum != 0)

            {

**ans->next = new ListNode(sum);**

**ans = ans->next;**

                sum = 0;

            }

            head = head->next;

        }

    }

**return ans\_head->next;**

}

**3. is palindrome:**

bool isPalindrome(string s)

    {

        int i = 0, j = s.size() - 1;

        while (i < j)

        {

            if (s[i++] != s[j--])

                return false;

        }

        return true;

    }

Call: if (isPalindrome(s.substr(i, j))) // s.substr is the library func;

//I , j is two nested loop ;

4. Longest continuous substring;

int longestContinuousSubstring(string s)

    {

        char prev = 'a';

        int cnt = 0,ans = 0;

        for(char c:s)

        {

            if(c==prev+1)

            {

                cnt++;

                prev = c;

            }

            else

            {

                cnt = 1;

                prev = c;

            }

            ans = max(ans,cnt);

        }

        return ans;

    }

**5.Tree :**

Node \*buildTree(Node \*root)

{

    int data;

    cout << "Enter the data: ";

    cin >> data;

    root = new Node(data);

    if (data == -1)

    {

        return NULL;

    }

    cout << "Enter the value for left data" << data << "-> ";

    root->left = buildTree(root->left);

    cout << "Enter the value for right data" << data << "-> ";

    root->right = buildTree(root->right);

    return root;

}

void levelOrderTraversal(TreeNode\* &root)

{

    queue<TreeNode\*>q;

    q.push(root);

    while(!q.empty())

    {

        TreeNode\* temp = q.front();

        cout<<temp->data<< " ";

        q.pop();

        if(temp->left !=NULL)

        {

            q.push(temp->left);

        }

        if(temp->right)

        {

            q.push(temp->right);

        }

    }

}

6. smart loop :

for(int i=0,j=v.size()-1 ; i<v.size()/2 && j>v.size()/2;i++,j--){

            if(v[i]+v[j]==k)

            {

                ans++;

            }

}

Time complexity is O(n);

**7. prefix Sum:**

void func(vector<int>v,int k){

        int preSum = v[0];

        for(int i=1; i<v.size();i++)

        {

           preSum += v[i];

           v[i] = preSum;

        }

        cout<<"Prefix Sum is :";

        for(int i=0; i<v.size();i++)

        {

           cout<<v[i]<<" ";

        }cout<<endl;

        // cout<<"sum of i,j is : ";

        // cout<<v[j]-v[i-1]; // i,j dhore niba or qsn a dewa thakbe mone koro;

    }

**8. BFS Traversal :**

First u need to take input as adjacency list:

**vector<int>vect[n];**

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

**{**

**vect[edges[i][0]].push\_back(edges[i][1]);**

**vect[edges[i][1]].push\_back(edges[i][0]);**

**}**

void bfsTraversal(vector<int>vect[],int sz,int source)

{

    queue<int>q;

    vector<int>visited(sz+1,false);

    visited[source] = true;

    q.push(source);

    while(!q.empty())

    {

        int vertex = q.front();

        cout<<vertex<<" ";

        q.pop();

        for(auto it:vect[vertex])

        {

            if(visited[it]==false)

            {

                visited[it]= true;

                q.push(it);

            }

        }

    }

}

**9. map greater version sort :**

int main()

{

    map<int, string, greater<int> > mymap;

    // Inserting the elements one by one

    mymap.insert(make\_pair(10, "queen"));

    mymap.insert(make\_pair(20, "rose"));

    mymap.insert(make\_pair(5, " lion"));

    // begin() returns to the first value of map

    map<int, string>::iterator it;

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

        cout << "(" << (\*it).first << ", " << (\*it).second

            << ")" << endl;

    return 0;

}

output:

(20, rose)

(10,queen)

(5 , lion)

**// Map sorting ascending order :**

vector<pair<string, int>> sortingMap (map<string,int>mapp){

    // create an empty vector of pairs

    vector<pair<string, int>> vec;

    // copy key-value pairs from the map to the vector

    copy(mapp.begin(),

         mapp.end(),

         back\_inserter<vector<pair<string, int>>>(vec));

    // sort the vector by increasing the order of its pair's second value

    // if the second value is equal, order by the pair's first value

    sort(vec.begin(), vec.end(),

         [](const pair<string, int> &l, const pair<string, int> &r)

    {

        if (l.second != r.second)

        {

            return l.second > r.second;  //change this < for decending value

        }

        return l.first < r.first;

    });

    return vec;

    /\*

    // print the vector

    for (auto const &pair: vec)

    {

        cout << '{' << pair.first << "," << pair.second << '}' << endl;

    }

    \*/

}

10. Set best technique:

**Input:** nums1 = [4,9,5], nums2 = [9,4,9,8,4]

**Output:** [9,4]

class Solution {

public:

    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {

        unordered\_set<int>s1(nums1.begin(),nums1.end());

        vector<int>ans;

        for(auto it : nums2)

        {

            if(s1.count(it)>=1)

            {

                ans.push\_back(it);

                s1.erase(it);

            }

        }

        return ans;

    }

};

// special: if(set.count(s[i])==0)

**11. Decimal to binary** : TC-> O(log n)

string DecimalToBinary(int num)

{

    string str;

    while (num)

    {

        if (num & 1) // 1

            str += '1';

        else // 0

            str += '0';

        num >>= 1; // Right Shift by 1

    }

    reverse(str.begin(),str.end());

    return str;

}

Second Technique: TC-> O(1)

int decToBinary(int n)

{

    for (int i = 31; i >= 0; i--) {

        int k = n >> i;

        if (k & 1)

            cout << "1";

        else

            cout << "0";

    }

}

**GCD and LCD:**

def gcd(a,b):

if a==0:

return b

return gcd(b%a,a)

def lcm(a,b):

gcd\_Receiver = gcd(a,b)

product = a\*b

return product//gcd\_Receiver

t = int(input("Enter the testCase: "))

while t!=0 :

n,m = map(int,input("Enter two values: ").split())

print("GCD output is: {}".format(gcd(n,m)))

print("LCM output is: {}".format(lcm(n,m)))

t-=1

**// pivot index;**

class Solution

{

public:

    int pivotIndex(vector<int>& nums)

    {

        int rightSum = 0;

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

        {

            rightSum+=nums[i]; // Total sum of the array

        }

        int leftSum = 0;

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

        {

            rightSum-=nums[i];

            if(rightSum==leftSum)

            {

                return i;

            }

            else

            {

                leftSum+=nums[i];

            }

        }

        return -1;

    }

};

int main()

{

    vector<int>v;

    for (int i=0; i<2; i++)

    {

        int x;

        cin>>x;

        v.push\_back(x);

    }

    Solution obj = Solution();

    int ans = obj.pivotIndex(v);

    cout<<ans;

}

**//setprecision**

int main()

{

    double a,b,c;

    double pi=3.14159,tri,cri,tra,qua,ret;

    cin>>a>>b>>c;//u can add fixed input precision like cin>>fixed>>setprecision(4)>>a;

tri =0.5\*a\*c;

    // fixed is used so that the all value is being fixed .. here 3 values is fixed.

    // every output must fulfill the fixed 3 postions .

    cout<<setprecision(3)<<"TRIANGULO: "<<tri<<endl;

    return 0;

}

**// loop i%something**

int main()

{

    int t;

    cin >> t;

    while (t--)

    {

        int n, a, b;

        cin >> n >> a >> b;

        for (int i = 0; i < n; ++i)

        {

            cout << char('a' + i % b); // best

// code name: construct string;

        }

        cout << endl;

    }

    return 0;

}

**// Binary search:**

class Solution

{

    public:

    int binarySearch1(vector<int> v, int target)

    {

        int left = 0, right = v.size() - 1, mid;

        int result = -1; // ans index value;

        while (left <= right)

        {

            mid = left + (right - left) / 2;

            if (v[mid] == target)

            {

                result = mid;

                right = mid - 1;

            }

            else if (v[mid] > target)

                right = mid - 1;

            else

                left = mid + 1;

        }

        return result;

    }

    int binarySearch2(vector<int> v, int target)

    {

        int left = 0, right = v.size() - 1, mid;

        int result = -1; // ans index value;

        while (left <= right)

        {

            mid = left + (right - left) / 2;

            if (v[mid] == target)

            {

                result = mid;

                left = mid + 1;

            }

            else if (v[mid] > target)

                right = mid - 1;

            else

                left = mid + 1;

        }

        return result;

    }

    vector<int> searchRange(vector<int> &v, int target)

    {

        //print 1st and last index of

        // an sorted array;

        vector<int> ans(2);

        ans[0] = binarySearch1(v, target);

        ans[1] = binarySearch2(v, target);

        return ans;

    }

};

**// construct the large number**

**// nums = [3,30,34,5,9]**

**// output: "9534330"**

class Solution

{

public:

    string largestNumber(vector<int>& v)

    {

        string ans = "";

        for(int i=0; i<v.size()-1; i++)

        {

            for(int j=i+1; j<v.size(); j++)

            {

                bool temp = swap\_able(v[i],v[j]);

                if(temp==true)

                {

                    swap(v[i],v[j]);

                }

            }

        }

        for(int i=0; i<v.size(); i++)

        {

            ans+= to\_string(v[i]);

        }

        return ans[0]=='0'? "0" : ans;

    }

    bool swap\_able(int a,int b)

    {

        string s1 = to\_string(a);

        string s2 = to\_string(b);

        return s1+s2 < s2+s1;

    }

};

**// productExceptSelf,index er porer sob product**

class Solution

{

public:

    vector<int> productExceptSelf(vector<int> v)

    {

        int n = v.size();

        vector<int> ans(n);

        ans[n - 1] = 1; // ans array used as right product array

        for (int i = n - 2; i >= 0; i--)

        {

            ans[i] = v[i + 1] \* ans[i + 1];

        }

        int leftProd = 1;

        for (int i = 0; i < n; i++)

        {

            ans[i] = leftProd \* ans[i];

            leftProd = leftProd \* v[i];

        }

        return ans;

    }

};

**// “12” + “13” = 25**

**// string addd;**

class Solution

{

public:

    string addStrings(string num1, string num2)

    {

        int i = num1.size()-1;   // num1 last last pointing index;

        int j = num2.size()-1;   // num2 last last pointing index;

        int carry = 0;

        string ans = "";

        while(i>=0 || j>=0)

        {

            int sum = carry;

            if(i>=0)

            {

                sum += num1[i]-'0';

                i--;

            }

            if(j>=0)

            {

                sum += num2[j] - '0';

                j--;

            }

            carry = sum > 9 ? 1 : 0;

            ans = ans + to\_string(sum%10);

        }

        if(carry == 1)

        {

            ans += to\_string(carry);

        }

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

        return ans;

    }

};

**//Subarray Sum Equals K**

class Solution {

public:

    int subarraySum(vector<int>& v, int k) {

        int preSum = v[0],ans=0;

        map<int,int>mp;

        for(int i=1; i<v.size();i++)

        {

           preSum += v[i];

           v[i] = preSum;

        }

        for(int i=0; i<v.size();i++)

        {

            if(v[i]==k)

                ans++;

            if(mp.find(v[i]-k)!=mp.end())

            {

                ans+=mp[v[i]-k];

            }

            mp[v[i]]+=1;

        }

        return ans;

    }

};

// for test case : 7  7  7  7

// prefix Sum    : 7  14 21 28

// mp[nums[i]] -> count

//     7       -> 1

//     14      -> 1

//     21      -> 1

//     28      -> 1

// **Subarray Sums Divisible by K**

class Solution

{

public:

    int subarraysDivByK(vector<int> &arr, int k)

    {

        map<int, int> m;

        int ans = 0;

        m.insert({0,1});

        int cumSum = 0;

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

        {

            cumSum+=arr[i];

            int remainder = cumSum % k;

            if(remainder < 0 )remainder+=k;

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

            {

                ans+= (m[remainder]);

                m[remainder]++;

            }

            else

            {

                m.insert({remainder,1});

            }

        }

        return ans;

    }

};

// Remove maximum digit from a number

class Solution {

public:

    string removeDigit(string number, char digit) {

       string ans = "";

    for(int i=number.size(); i>=0; i--)

    {

        if(number[i]==digit)

        {

            string temp = number.substr(0,i) + number.substr(i+1,number.size());

            ans = max(ans,temp);

        }

    }

        return ans;

    }

};

**number = "123", digit = "3"**

**output: ‘12’;**

// Graph: Find if path exist or not;

class Solution {

public:

    bool validPath(int n, vector<vector<int>>& edges, int source, int destination) {

        vector<int>vect[n];

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

        {

            vect[edges[i][0]].push\_back(edges[i][1]);

            vect[edges[i][1]].push\_back(edges[i][0]);

        }

        queue<int>q;

        vector<int>visited(n,false);

        visited[source] = true;

        q.push(source);

        while(!q.empty())

        {

            int vertex = q.front();

            //cout<<vertex<<" ";

            q.pop();

            for(auto it:vect[vertex])

            {

                if(visited[it]==false)

                {

                    visited[it]= true;

                    q.push(it);

                }

            }

        }

        return visited[destination];

    }

};

**// missing number from an sorted array;**

class Solution {

public:

    int missingNumber(vector<int>& nums) {

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

        for(int i=nums.size()-1;i>0;i--)

        {

            if(nums[i]-nums[i-1] !=1){

                return nums[i]-1;

            }

        }

        if(nums[0]==0){

            return nums[nums.size()-1]+1;

        }

        else{

            return 0;

        }

    }

};

// Binary search lower bound;

int main()

{

    vector<int> arr1 = { 10, 15, 20, 25, 30, 35};

    vector<int> arr2 = { 10, 15, 20, 20, 25, 30, 35 };

    vector<int> arr3 = { 10, 15, 25, 30, 35 };

    cout << "The position of 20 using lower\_bound "

            " (in single occurrence case) : ";

    cout << lower\_bound(arr1.begin(), arr1.end(), 20)

                - arr1.begin();

    cout << endl;

    // using lower\_bound() to check if 20 exists

    // multiple occurrence

    // prints 2

    cout << "The position of 20 using lower\_bound "

            "(in multiple occurrence case) : ";

    cout << lower\_bound(arr2.begin(), arr2.end(), 20)

                - arr2.begin();

    cout << endl;

    // using lower\_bound() to check if 20 exists

    // no occurrence

    // prints 2 ( index of next higher)

    cout << "The position of 20 using lower\_bound "

            "(in no occurrence case) : ";

    cout << lower\_bound(arr3.begin(), arr3.end(), 20)

                - arr3.begin();

    cout << endl;

}

The position of 20 (in single occurrence case) : 2

The position of 20 (in multiple occurrence case) : 2

The position of 20 (in no occurrence case) : 2

// Upper bound

int main()

{

    vector<int> arr1 = { 10, 15, 20, 25, 30, 35 };

    vector<int> arr2 = { 10, 15, 20, 20, 25, 30, 35 };

    vector<int> arr3 = { 10, 15, 25, 30, 35 };

    // using upper\_bound() to check if 20 exists

    // single occurrence

    // prints 3

    cout << "The position of 20 using upper\_bound"

            " (in single occurrence case) : ";

    cout << upper\_bound(arr1.begin(), arr1.end(), 20)

                - arr1.begin();

    cout << endl;

    // using upper\_bound() to check if 20 exists

    // multiple occurrence

    // prints 4

    cout << "The position of 20 using upper\_bound "

            "(in multiple occurrence case) : ";

    cout << upper\_bound(arr2.begin(), arr2.end(), 20)

                - arr2.begin();

    cout << endl;

    // using upper\_bound() to check if 20 exists

    // no occurrence

    // prints 2 ( index of next higher)

    cout << "The position of 20 using upper\_bound"

            " (in no occurrence case) : ";

    cout << upper\_bound(arr3.begin(), arr3.end(), 20)

                - arr3.begin();

    cout << endl;

}

The position of 20 (in single occurrence case) : 3

The position of 20 (in multiple occurrence case) : 4

The position of 20 (in no occurrence case) : 2

**Binary search library func:**

if (binary\_search(arr.begin(), arr.end(), 15))

cout << "15 exists in vector";

Ascending order sort:

sort(arr,arr+n);

1,2,3,4,5;

Descending order sort:

sort(arr,arr+n,greater<int>());

sort(v.begin(),v.end(),greater<int>());

size of an array:

sizeof(arr)/sizeof(arr[0]);

max&min element of an array:

cout<<\*max\_element(arr,arr+n);

cout<<\*min\_element(arr,arr+n);

Reverse an array:

reverse(arr,arr+n);

Clear an array:

memset(arr,0,sizeof(arr));

Auto size and input:

sring s(n,’a’);

int to char => int\_value + ‘0’;

to\_string(num);

char to int => char\_value – ‘0’;

stoi(string);

int\_max = 2147483647

int\_min = -2147483648

setname.clear();

setname.erase(element);

1+2+3+4+….+n = {n(n+1)}/2;

1^2+2^2+3^2+….+n^2 = {n(n+1) \*( 2n+1)}/6;

1^3+2^3+3^3+….+n^3 = [{n(n+1)}/2]^2;