C++ Template

```cpp
/* TLE?
#pragma GCC optimize("Ofast,unroll-loops")//-0.0+0.0=-0
#pragma GCC target("avx2") // STL cointaner = std::allocator err
*/
#include<bits/stdc++.h>

using ull = unsigned long long;
using ll = long long;
using namespace std;
#define endl '\n'
#define all(x) (x).begin(), (x).end() // sort( all(vec) );
#define yn(x) (cout << ((x) ? "YES" : "NO"))
#define dbg(...) cerr<<"LINE("<<__LINE__<<")->["<<#__VA_ARGS__<<"]:<<(__VA_ARGS__);
#define gs(n) ((n * (n + 1)) >> 1)
#define pb push_back
#define F first
#define S second

int main(){
ios_base::sync_with_stdio(0);cin.tie(0);
//freopen("in.txt", "r", stdin);
    int tc=2;
    //cin>>tc;
    int n;
    while(tc--){
        cin>>n;
        for(int i=0;i<n;++i){
            //code
        }
    }
return 0;
}
```

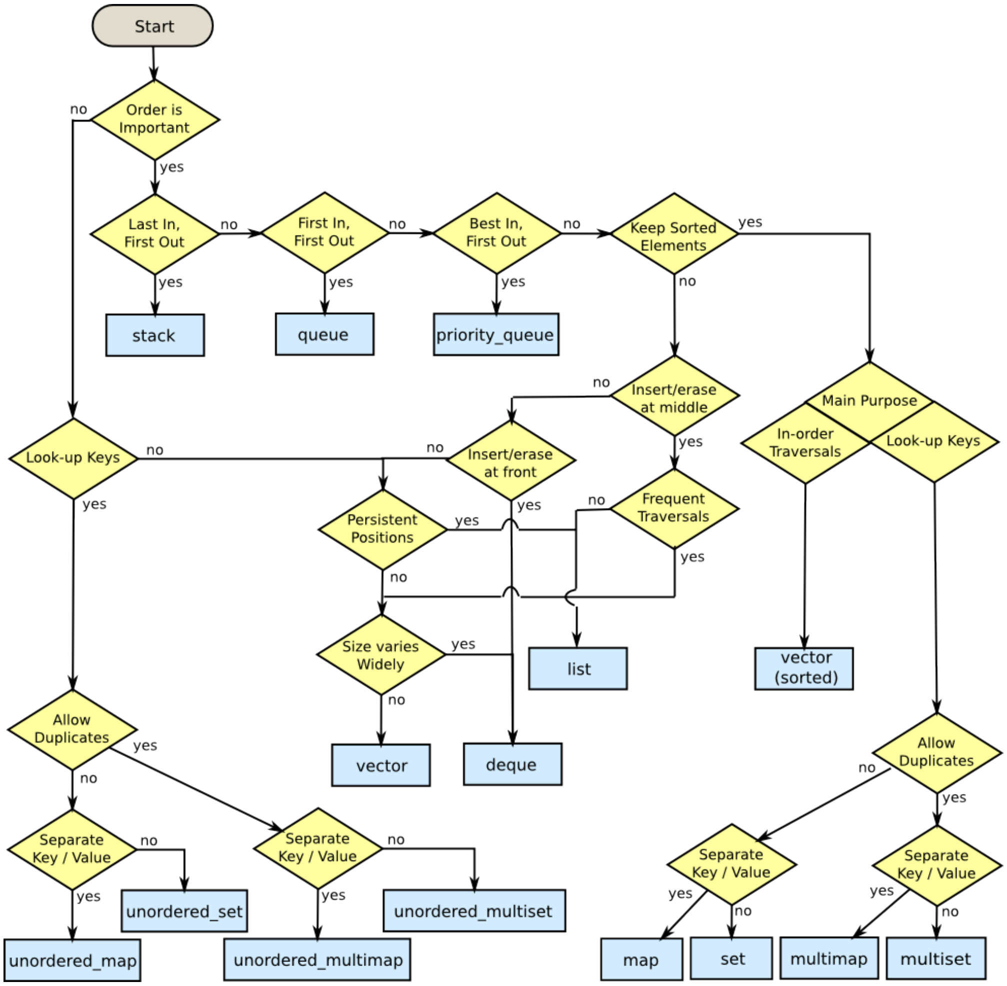
Primes

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

```cpp
void sieve_of_eratosthenes(int n,vector<bool>& is_prime) {
    is_prime[0] = is_prime[1] = false;

    for (int p = 2; p * p <= n; ++p) {
        if (is_prime[p]) {
            for (int i = p * p; i <= n; i += p) {
                is_prime[i] = false;
            }
        }
    }
}
```

STL Containers



Primitive data types (x64)

(-10^9 to +10^9)
int: -2,147,483,648 to 2,147,483,647
unsigned int: 0 to 4,294,967,295

(-10^18 to +10^18)
long long: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long: 0 to 18,446,744,073,709,551,615

(7 digit precision):
float: 1,175494351 E - 38 to 3,402823466 E + 38

(15 digit precision):
double: 2,2250738585072014 E - 308 to 1,7976931348623158 E + 308

(18–19 digit precision):
long double: 3.3621031431120935063 E1-4932 to 1.1897314953572317650 E 4932. (slow)

Misc Math functions

```
min({a, b, c, d});  max({a, b, c, d});
bool flag; cout << (flag ? "YES" : "NO");//ternary op
numeric_limits<unsigned int>::max();  numeric_limits<float>::min();
cout << fixed << setprecision(digits)<< var;// digits == 0 ? (round)
round(num);//1.45 -> 1 , 1.5 -> 2
trunc(num);//1.5 -> 1
ceil(num);//1.5 -> 2 ; int-ceil `a/b` -> `(a + b - 1) / a`
floor(num);//1.5 -> 1
abs(num);// -1.5 -> 1.5, 1.5 -> 1.5
sqrt(num);  sqrtl(num);
pow(base, exp); powl(base,exp); pow(p, 1.0 / n);// nth root of p
numeric_limits<double>::infinity(); isinf();
std::numeric_limits<double>::quiet_NaN(); isnan( sqrt(-1) );
gcd(a,b);// __gcd(a,b);// (c++ < 17)
lcm(a,b);// ( (a * b ) / __gcd(a,b) );
```

Algorithms

```
bool isPrime(int n){
    if (n<2) return false;
    if (n<=3) return true;
    if (!(n%2) || !(n%3)) return false;
    for (int i=5;i*i<=n;i+=6)
        if (!(n%i) || !(n%(i+2))) return false;
    return true;
}
int binarySearch(vector<int> v, int l, int r, int x){
    while (l <= r) {
        int m = l + (r - l) / 2;
        if (v[m] == x)
            return m;
        if (v[m] < x)
            l = m + 1;
        else
            r = m - 1;
    }
    return -1;
}
```

Ciruclar Array

```
int dist1 = abs(idx1-idx2);         int dist2 = n - dist1;
for (int i = idx1; i != idx2; i = (i - 1 + n) % n) {}//Left
for (int i = idx1; i != idx2; i = (i + 1 + n) % n) {}//right
```

sorting

```
sort(vec.begin(), vec.end());//ascending
sort(vec.begin(), vec.end(), greater<int>());//non ascending

inline bool myOrder(pair<int, int> p1, pair<int, int> p2) {
    return p1.first < p2.first;
}
vector<pair<int, int>> vec = {{3, 1}, {2, 5}, {1, 4}};
sort(vec.begin(), vec.end(), myOrder);
for(auto elem : vec)
    cout << "(" << elem.first << ", " << elem.second << ") ";
```

Standard Template Library (STL)

Iterators

```
for(vector<int>::iterator it=stl.begin(); it!=stl.end(); it++)
    cout<<*it<<" ";//(it) is memory pointer (*it) its value
for (const auto& pair : mapVar)
    cout << pair.first << ": " << pair.second;
.rend .rbegin
```

Capacity:

```
stl.size(); //lenght of STL container
stl.empty(); // boolean function
```

Modifiers:

```
stl.clear();//clears the container but complexity O(n) better use swap(v,vv[i])
stl.erase(it);stl.erase(itBegin,itEnd);// removes elements from  range
stl.resize(size);stl.resize(size,initializer);stl.insert(it,elem)
stl.emplace_back(elem)//faster push_back (avoid copies), inmediate values
```

.fun()	.push_back(elem)	.pop_back()	.insert(elem)	.find(elem)
vector	x	x		find(v.beg(),v.end(),elem)
string	x	x	x	
maps			(make_pair(e,e))	x
sets			(elem)	x

Operations

```
if (stl.find(elem) != stl.end())//found
stl.count(elem);//count the occurrences of the element, sets and maps only
```

STL Algorithms

```
int sum = accumulate(v.begin(), v.end(), sumStart);
int product = accumulate(v.begin(), v.end(), 1,multiplies<int>());
merge(v1.begin(), v1.end(), v2.begin(), v2.end(), back_inserter(vecM));
fill(v.begin(), v.end(), value);iota(vec.begin(), vec.end(),start);
auto maxElement = max_element(vec.begin(), vec.end());
// use: cout << (maxElement != vec.end() ? *maxElement : “NO”);
```

strings

```
string substr1=str1.substr(start, end);
int found1 = str.find(substr1);//first find at
int found2 = str.find(substr1, found1 + 1);//second find at
if (found1 != string::npos) //found

str1.replace(start, str2.size(), str2);
str1.erase(start, end);
str1.append(str2); str3 = str1 + str2;

char ch;
isalpha(ch);isdigit(ch);isupper(ch);isler(ch);//char bools
toler(ch);toupper(ch);//alphabet
char numC='1'; int  numI=numC-'0';//toInt
int numI=1;   char numC=numI+'0';//toChar
int a=stoi(str); ull b=stoull(str); double c = stod(str)// str to nums
str = to_string(a); //int to str

std::stringstream ss;    ss << str1.substr(0, 1) << str1.substr(3, 4); ss.str();
```

circular string rotation

```
string s;
vector<string> v1(s.size());
vector<string> v2(s.size());
for(int i=0;i<s.size();i++){
    v1[i] = s.substr(i,s.size()-i) + s.substr(0,i);//left
    v2[i] = s.substr(s.size()-i,i) + s.substr(0,s.size()-i);//right
}
```

Bitmasking

```
for (int i = 0; n > 0 ; ++i) vecBin.push_back( (n >> i) & 1 );// int2bin
var<<exp;//var*(2^exp)
var>>exp;//var/(2^exp)
bitset<8> decBset(8); bitset<8> strBset("1100"); bitset<8> binBset(0b001);

decBset.set(4);        // Set the bit at idx 4 to 1
decBset.reset(4);      // Reset the bit at idx 4 to 0
decBset.flip(0);       // Flip the bit at idx 0 (0 becomes 1, and 1 becomes 0)

int numSetBits = decBset.count();  // Count the number of bits that are set to 1
bool bit2IsSet = decBset.test(2);  // Check if the bit at idx 2 is set to 1
bool anyBSet = decBset.any();       // Check if any bit is set to 1
bool noBSet = decBset.none();       // Check if no bits are set to 1
bool allBSet = decBset.all();       // Check if all bits are set to 1


int bsetSize = decBset.size();
string bsetStr = decBset.to_string();
unsigned long bsetULong = decBset.to_ulong();
unsigned long long bsetULLong = decBset.to_ullong();

if(num & 1) // odd
else // even

__builtin_popcount(x);//Counts the number of one’s(set bits) in an integer
__builtin_parity(x);//Checks the Parity of a number.Returns true(1) if the number has odd
parity(odd number of set bits) else (0)
__builtin_clz(x);// Counts the leading number of zeros of the integer
__builtin_ctz(x);// Counts the trailing number of zeros of the integer
__builtin_ffs(x)// (Find First Set) returns the index of the least significant bits of x+1
__lg(x);// returns the index of the highest set bit.
```

DP

```
Devuelve el el minimo de los maximos entre pares de rangos consecutivos haciendo cortes en
el Array.

const int MAX = 1005;
ll dp[MAX][MAX];
ll sum_ran[MAX][MAX];
int N;

ll f(int i, int cuts) {
    if (cuts == 0) return sum_ran[i][N-1];
    if (i == N) return 0;
    ll &ans = dp[i][cuts];
    if (ans != - 1) return ans;
    for (int j = i; j < N; j++) {
        ans = min(ans, max(sum_ran[i][j], f(i + 1, cuts - 1)));
    }
}
```

```
// METODO PARA CALCULAR EL LIS en O(n^2) y O(nlog(n)). La ventaja de tener a mano O(n^2) es porque es mas facil de codear, entender y modificar

const int MAX = 1e5+1;
int A[MAX];
int dp[MAX];
int N = MAX;
vector<int> LIS; // PARA Lis_opt

// LIS O(nlog(n)) Para longest non-decreasing cambiar lower_bound por upper_bound
int lis_opt() {
    LIS.clear();
    for (int i = 0; i < N; i++) {
        auto id = lower_bound(LIS.begin(), LIS.end(), A[i]);
        if (id == LIS.end()) {
            LIS.pb(A[i]);
            dp[i] = LIS.size();
        }
        else {
            int idx = id - LIS.begin();
            LIS[idx] = A[i];
            dp[i] = idx + 1;
        }
    }
    return LIS.size();
}

// METODO PARA RECONSTRUIR LIS. Para non-decreasing cambiar < por <=
stack<int> rb;
void build() {
    int k = LIS.size();
    int cur = oo;
    for (int i = N - 1; i >= 0, k; i--) {
        if (A[i] < cur && k == dp[i]) {
            cur = A[i];
            rb.push(A[i]);
            k--;
        }
    }
}
```

Dada una lista de enteros, retorna la máxima suma de un rango de la lista.

```
int maxRangeSum(vector<int> a) {
    int sum = 0, ans = 0;
    for (int i = 0; i < a.size(); i++) {
        if (sum + a[i] >= 0) {
            sum += a[i];
          ans = max(ans, sum);
        } else sum = 0;
    } return ans;
}
```

Devuelve el el minimo de los maximos entre pares de rangos consecutivos haciendo cortes en el Array.

```
const int MAX = 1005; ll dp[MAX][MAX]; ll sum_ran[MAX][MAX]; int N;
ll f(int i, int cuts) {
    if (cuts == 0) return sum_ran[i][N-1];
    if (i == N) return 0;
    ll &ans = dp[i][cuts];
    if (ans != - 1) return ans;
    for (int j = i; j < N; j++)
        ans = min(ans, max(sum_ran[i][j], f(i + 1, cuts - 1)));
}
```

Problema del viajero. Devuelve la ruta minima haciendo un tour visitando todas los nodos (ciudades) una unica vez.

```cpp
const int MAX = 18;
int target; // Inicializarlo para (1<<N) - 1
int dist[MAX][MAX]; // Distancia entre cada par de nodos
int N;
int dp[(1<<MAX) + 2][MAX];
vector<int> rb;
const int INF = (int) (2e9);

// Llamar para TSP(0, -1) Si no empieza de ninguna ciudad especificia
// De lo contrario llamar TSP(0, 0)
int TSP(int mask, int u) {
    if (mask == target) {
        return 0;
        // O en su defecto el costo extra tras haber visitado todas las ciudades. EJ: Volver a la ciudad principal
    }
    if (u == -1) {
        int ans = INF;
        for (int i = 0; i < N; i++) {
            ans = min(ans, TSP(mask | (1<<i), i));
            // Agregar costo Extra desde el punto de partida si es necesario
        }
        return ans;
    }
    int &ans = dp[mask][u];
    if (ans != -1) return ans;
    ans = INF;
    for (int i = 0; i < N; i++) {
        if (!(mask & (1<<i)))
            ans = min(ans, TSP(mask | (1<<i), i) + dist[u][i]);
    }
    return ans;
}

void build(int mask, int u) {
    if (mask == target) return; // Acaba el recorrido
    if (u == -1) {
        for (int i = 0; i < N; i++) {
            if (TSP(mask, u) == TSP(mask | (1<<i), i)) {
                rb.pb(i);
                build(mask | (1<<i), i);
                return;
            }
        }
    } else {
        for (int i = 0; i < N; i++) {
            if (!(mask & (1<<i))) {
                if (TSP(mask, u) == TSP(mask | (1<<i), i) + dist[u][i]) {
                    rb.pb(i);
                    build(mask | (1<<i), i);
                    return;
                }
            }
        }
    }
}
```

Algoritmos varios

```
//a^b mod p
long powmod(long base, long exp, long modulus) {
    base %= modulus;
    long result = 1;
    while (exp > 0) {
        if (exp & 1) result = (result * base) % modulus;
        base = (base * base) % modulus;
        exp >>= 1;
    }
return result;
```

```
Factorial mod:  n! mod p
int factmod (int n, int p) {
long long res = 1;
while (n > 1) {
    res = (res * powmod (p-1, n/p, p)) % p;
    for (int i=2; i<=n%p; ++i)
        res=(res*i) %p;
        n /= p;
    }
return int (res % p);
}
```

```
Generate combinations
// n>=m, choose M numbers from 1 to N.
void combination(int n, int m) {
    if (n<m) return ;
    int a[50]={0};
    int k=0;
    for (int i=1;i<=m;i++) a[i]=i;
    while (true) {
        for (int i=1;i<=m;i++)
            cout << a[i] << " ";
        cout << endl;
        k=m;
        while ((k>0) && (n-a[k]==m-k)) k--;
        if (k==0) break;
        a[k]++;
        for (int i=k+1;i<=m;i++)
            a[i]=a[i-1]+1;
    }
}
```