C++ Template and Common Data Structures

Basic Template

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    return 0;
}
```

Vector Operations

```cpp
// Input vector
vector<int> v(n); for(int i = 0, i < n, i++) cin >> v[i];

// Print vector
for(int x : v) cout << x << " "; cout << endl;

// Common operations
v.size();                   // Size
v.empty();                  // Is empty?
v.push_back(x);             // Add to end
v.pop_back();               // Remove last
v.erase(v.begin()+i);       // Erase at index i
v.insert(v.begin()+i,x);    // Insert x at index i
sort(v.begin(),v.end());    // Sort ascending
sort(v.rbegin(),v.rend());  // Sort descending
reverse(v.begin(),v.end()); // Reverse
*min_element(v.begin(),v.end()); // Min element
*max_element(v.begin(),v.end()); // Max element
binary_search(v.begin(),v.end(),x); // Check if x exists
lower_bound(v.begin(),v.end(),x); // First >= x
upper_bound(v.begin(),v.end(),x); // First > x
```

Common Algorithms

```cpp
// GCD
long long gcd(long long a, long long b) { return b ? gcd(b, a%b) : a;
  }

// Sieve of Eratosthenes
vector<bool> isPrime(n+1,true);
void sieve(int n) {
    isPrime[0] = isPrime[1] = false;
    for(int i=2;i*i<=n;++i)
        if(isPrime[i])
            for(int j=i*i;j<=n;j+=i)
                isPrime[j] = false;
}

// Factorial
long long factorial(int n) {
    long long res = 1;
    for(int i = 1; i <= n; i++) res *= i;
    return res;
}

// Sum of digits
int sumOfDigits(long long n) {
    int sum = 0;
    while (n > 0) {
        sum += n % 10;
        n /= 10;
    }
    return sum;
}
```

Set Operations

```cpp
set<int> s;           // Init set
s.insert(x);          // Add x
s.erase(x);           // Remove x
s.count(x);           // Check if x exists
s.find(x)!=s.end();   // Alternative check
for(int x : s) cout << x << " "; // Print set
```

Map Operations

```cpp
map<string,int> m;    // Init map
m[key] = val;         // Add/update key-val
m.count(key);         // Check if key exists
m.erase(key);         // Remove key
for(auto &p : m) cout << p.first << ":" << p.second << endl; // Print
   map
```