📑 Content List:

Fast Input Output Setup

```
#include <iostream>
using namespace std;
#define endl '\n'

void fastIO() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
}
```

Useful Math Functions

```
// GCD
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

// LCM
int lcm(int a, int b) {
    return (a / gcd(a, b)) * b;
}

// Modular addition
int modAdd(int a, int b, int
mod) {
    return ((a % mod) + (b %
mod)) % mod;
}

// Modular multiplication
int modMul(int a, int b, int
mod) {
    return ((a % mod) * (b %
mod)) % mod;
}
```

Check even or odd

```
bool isEven(int n) {
    return (n % 2 == 0);
}
```

Basic Loop & Control

```
// For loop
for (int i = 0; i < n; i++) {
    // code
}

// While loop
int i = 0;
while (i < n) {
    // code
    i++;
}

// If-else
if (condition) {
    // code
} else {
    // code
}

// Switch Case
switch (x) {
    case 1:
        // code
        break;
    case 2:
        // code
        break;
    default:
        // code
}
```

Sum of digits of a number

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

Sorting Techniques

```
// Bubble Sort
void bubbleSort(int arr[], int
n) {
    for (int i = 0; i < n-1;
i++) {
        for (int j = 0; j <
n-i-1; j++) {
            if (arr[j] >
arr[j+1]) {
                swap(arr[j],
arr[j+1]);
            }
        }
    }
}
```

Prime Checking & Power

```
// Check Prime
bool isPrime(int n) {
    if (n <= 1) return false;
    for (int i = 2; i*i <= n;
i++) {
        if (n % i == 0) return
false;
    }
    return true;
}

// Power Calculation
int power(int a, int b) {
    int result = 1;
    while (b > 0) {
        result *= a;
        b--;
    }
    return result;
}
```

Searching Techniques

```
// Linear Search
int linearSearch(int arr[], int
n, int key) {
    for (int i = 0; i < n; i++)
{
        if (arr[i] == key)
return i;
    }
    return -1;
}

// Binary Search (Array must be
sorted)
int binarySearch(int arr[], int
n, int key) {
    int low = 0, high = n-1;
    while (low <= high) {
        int mid = (low +
high)/2;
        if (arr[mid] == key)
return mid;
        else if (arr[mid] < key)
low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
```

Count digits in a number

```
int countDigits(int n) {
    int cnt = 0;
    while (n > 0) {
        cnt++;
        n /= 10;
    }
    return cnt;
}
```

Other Important Snippets

```
// Swap Two Numbers
void swapNumbers(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}

// Reverse an array
void reverseArray(int arr[], int
n) {
    int i = 0, j = n-1;
    while (i < j) {
        swap(arr[i], arr[j]);
        i++;
        j--;
    }
}

// Find Maximum element
int findMax(int arr[], int n) {
    int mx = arr[0];
    for (int i = 1; i < n; i++)
{
        if (arr[i] > mx) mx =
arr[i];
    }
    return mx;
}

// Find Minimum element
int findMin(int arr[], int n) {
    int mn = arr[0];
    for (int i = 1; i < n; i++)
{
        if (arr[i] < mn) mn =
arr[i];
    }
    return mn;
}
```

Basic Graph Templates

```
// BFS (Unweighted Graph)
#include <queue>
vector<int> adj[100];
bool visited[100];

void bfs(int start) {
    queue<int> q;
    q.push(start);
    visited[start] = true;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int v : adj[u]) {
            if (!visited[v]) {
                visited[v] =
true;
                q.push(v);
            }
        }
    }
}

// DFS (Unweighted Graph)
void dfs(int u) {
    visited[u] = true;
    for (int v : adj[u]) {
        if (!visited[v]) {
            dfs(v);
        }
    }
}
```

Factorial of a number

```
int factorial(int n) {
    int res = 1;
    for (int i = 2; i <= n; i++)
{
        res *= i;
    }
    return res;
}
```

Check palindrome

```
bool isPalindrome(int n) {
    int original = n, reversed =
0;
    while (n > 0) {
        reversed = reversed*10 +
n%10;
        n /= 10;
    }
    return original == reversed;
}
```

Sum of array elements

```
int sumArray(int arr[], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++)
{
        sum += arr[i];
    }
    return sum;
}
```

Find second largest

```
int secondMax(int arr[], int n)
{
    int first = INT_MIN, second
= INT_MIN;
    for (int i = 0; i < n; i++)
{
        if (arr[i] > first) {
            second = first;
            first = arr[i];
        }
        else if (arr[i] > second
&& arr[i] < first) {
            second = arr[i];
        }
    }
    return second;
}
```

Simple Sieve for primes

```
// Mark all primes up to N
bool isPrime[N+1];

void sieve(int N) {
    for (int i = 0; i <= N; i++)
isPrime[i] = true;
    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;
            }
        }
    }
}
```

Count element frequency

```
#include <unordered_map>
void countFrequency(int arr[],
int n) {
    unordered_map<int, int>
freq;
    for (int i = 0; i < n; i++)
{
        freq[arr[i]]++;
    }
    for (auto it = freq.begin();
it != freq.end(); ++it) {
        cout << it->first << ":
" << it->second << endl;
    }
}
```

Count even and odd

```
void countEvenOdd(int arr[], int
n) {
    int even = 0, odd = 0;
    for (int i = 0; i < n; i++)
{
        if (arr[i] % 2 == 0)
even++;
        else odd++;
    }
    cout << "Even: " << even <<
", Odd: " << odd << endl;
}
```

Bonus: Macros

```
#define ll long long
#define pb push_back
#define F first
#define S second
```