Contents

Basics

C++

Template

```cpp
#include <bits/stdc++.h> //Include all c++ libraries
using namespace std;
#define io_op
↪  ios_base::sync_with_stdio(false);cin.tie(NULL);
↪  //IO Optimation
#define llli __int128_t
#define lli long long int
#define li long int
#define ld long double
#define pii pair<int, int>
#define pll pair<lli, lli>
#define m_p make_pair
#define fi first
#define se second
#define vint vector<int>
#define vli vector<li>
#define vlli vector<lli>
#define vpii vector<pii>
#define vpll vector<pll>
#define p_b push_back
#define all(v) v.begin(), v.end()
#define alla(arre, size) arre, arre + size
#define forv(v, i) for(int i=0; i<v.size(); i++)
↪  //For Vector
#define rforv(v, i) for(int i=v.size()-1; i>-1; i--)
↪  //For Reverse vector
#define forx(x, i) for(int i=0; i<x; i++) //For Number
#define showv(v, i) for(auto i:v) cout << i << ' ';
↪  //Display vector
#define npermute next_permutation
#define ppermute prev_permutation

int main(){
        io_op
        return 0;
}
```

Output format

```cpp
int main() {
  ios state(nullptr);
  state.copyfmt(cout); //Saves current format state
  double D = 13.34567;
  cout << setprecision(4) << D << endl // 13.35
  << fixed << D << endl; // 13.3457
  cout.copyfmt(state); //Restores format
  int N = 13;
  cout << setw(4) << N << endl //" 13"
  << setfill('0') << N << endl //"0013"
  << left << N << endl // "1300"
  << right << N << endl // "0013"
  << hex << N << endl // "000d"
  << uppercase << N << endl // "000D"
  << nouppercase << N << endl // "000d"
  << oct << N << endl // "0015"
  << dec << N << endl; // "0013"
}
```

Permutations

```cpp
#define Dt int //Datatype int,long long,string,etc.
#define T vector<Dt>;

vector<T> permutations(T v) {
        vector<T> ans;
        sort(v.begin(), v.end());
        do ans.push_back(v);
        while (next_permutation(v.begin(), v.end()));
        return ans;
}
```

Priority Queue

```cpp
struct Object {
  int x, y;
};

int main() {
  // Comparison function cmp for objects
  auto cmp = [](const Object &a, const Object &b) {
    return a.x > b.x;
  };
  // Comparison function cmpd for datatypes
  auto cmpd = [](const int &a, const int &b) {
    return a > b;
  };
  // For object
  priority_queue<Object, vector<Object>,
                 decltype(cmp)> pq(cmp);
  // For datatypes
  priority_queue<int, vector<int>,
                 decltype(cmpd)> pq(cmpd);
}
```

Random

```cpp
mt19937_64 seed(chrono::steady_clock::now()
                    .time_since_epoch()
                    .count());

int random(int min, int max) {  // [min, max]
  return uniform_int_distribution<int>(min,
                                       max)(seed);
}

double random(double min, double max) {  // [min, max)
```

```
  return uniform_real_distribution<double>(min,
                                           max)(seed);
}
```

Read Line

```
// if mixing 'cin' with 'getline', remember to put
↪  'ignore' between 'cin' and 'getline' in that order

int main(){
  string s;
  getline(cin, s);
}
```

Set of Object

```
struct Object {
  int x, y;
};

int main() {
  // Comparison function cmp for objects
  auto cmp = [](const Object &a, const Object &b) {
    return a.x > b.x;
  };
  // Comparison function cmpd for datatypes
  auto cmpd = [](const int &a, const int &b) {
    return a < b;
  };
  //For object
  set<Object, decltype(cmp)> pq(cmp);
  //For datatypes
  set<Object, decltype(cmpd)> pq(cmpd);
}
```

Sort Vector of Objects

```
struct Object {
  char first;
  int second;
};

bool cmp(const Object &a, const Object &b) {
  return a.second > b.second;
}

int main() {
  vector<Object> v = {{'c', 3}, {'a', 1}, {'b', 2}};
  sort(v.begin(), v.end(), cmp);
}
```

Sort Vector of Pairs

```
bool sortbysec(const pair<int,int> &a,
               const pair<int,int> &b) {
    return a.second < b.second;
}
vector<pair<int, int>> pairs;
//Sort by first element
sort(pairs.begin(), pairs.end());
//Sort by second element
sort(pairs.begin(), pairs.end(), sortbysec);
```

String Split

```
vector<string> split(string str, char token) {
  stringstream ss(str);
  vector<string> v;
  while (getline(ss, str, token)) v.push_back(str);
  return v;
}
```

String to Int and Int to String

```
int main() {
  // String to Int
  int n = stoi("123") + 1;
  cout << n << endl;  // output: 124
  // stoll for long long int
  // stoull for unsigned long int
  // stod for double
  // stold for long double

  // Int to String
  // to_string converts int, double, long long int,
  ↪  etc. to string
  string str = "str+" + to_string(123 + 1);
  cout << str << endl;  // output: str+124
  return 0;
}
```

Substring

```
// [l, r)
string substr(string &s, int l, int r) {
  return string(s.begin() + l, s.begin() + r);
}
```

Python

Combinations

```
import itertools
# from arr choose k = > combinations(arr, k)
# example arr=[1, 2, 3] and k=3
print(list(itertools.combinations([1, 2, 3], 3)))
```

Fast IO

```
from sys import stdin, stdout

N = 10
# Reads N chars from stdin(it counts '\n' as char)
stdin.read(N)
# Reads until '\n' or EOF
line = stdin.readline()
# Reads all lines in stdin until EOF
lines = stdin.readlines()
# Writes a string to stdout, it doesn't add '\n'
stdout.write(line)
# Writes a list of strings to stdout
stdout.writelines(lines)
# Reads numbers separated by space in a line
numbers = list(map(int, stdin.readline().split()))
```

Permutations

```
import itertools
# All permutations of an arr
# Example arr=[1, 2, 3]
print(list(itertools.permutations([1, 2, 3])))
```

Random

```
import random
# Initialize the random number generator.
random.seed(None)
# Returns a random integer N such that a <= N <= b
random.randint(a, b)
# Returns a random integer N such that 0 <= N < b
random.randrange(b)
# Returns a random integer N such that a <= N < b
random.randrange(a, b)
# Returns an integer N with k random bits.
random.getrandbits(k)
# Shuffles a list
random.shuffle(li)
```

Sort List

```
li = ['a', 'c', 'b']
# sorts inplace in descending order
li.sort(reverse=True)
# returns sorted list ascending order
ol = sorted(li)
```

Sort List of Objects

```
class MyObject :
  def __init__(self, first, second, third):
    self.first = first
    self.second = second
    self.third = third

li = [MyObject('b', 3, 1), MyObject('a', 3, 2),
↪  MyObject('b', 3, 3)]
# returns list sorted by first then by second then by
↪  third in increasing order
ol = sorted(li, key = lambda x: (x.first, x.second,
↪  x.third), reverse=False)
# sorts inplace by first then by second then by third
↪  in increasing order
li.sort(key = lambda x: (x.first, x.second, x.third),
↪  reverse=False)
```