Build System for Sublime Text
Ubuntu - Linux:

```
{
"cmd" : ["g++ -std=c++14 $file_name -o $file_base_name && timeout 4s
./$file_base_name<inputf.in>outputf.in"],
"selector" : "source.c",
"shell": true,
"working_dir" : "$file_path"
}
```

Windows:

```
{
"cmd": ["g++.exe","-std=c++14", "${file}", "-o", "${file_base_name}.exe", "&&" ,
"${file_base_name}.exe<inputf.in>outputf.in"],
"selector":"source.cpp",
"shell":true,
"working_dir":"$file_path"
}
```

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

typedef long long ll;
typedef vector<int> vi;
typedef vector<ll> vl;
typedef vector<vi> vvi;
typedef vector<vl> vvl;
typedef pair<int,int> pii;
typedef pair<double, double> pdd;
typedef pair<ll, ll> pll;
typedef vector<pii> vii;
typedef vector<pll> vll;
typedef double dl;

#define PB push_back
#define F first
#define S second
#define MP make_pair
#define endl '\n'
#define all(a) (a).begin(),(a).end()
#define sz(x) (int)x.size()
#define mid(l,r) ((r+l)/2)
#define left(node) (node*2)
#define right(node) (node*2+1)
#define mx_int_prime 999999937
#define mod 1000000007

const double PI = acos(-1);
const double eps = 1e-9;
const int inf = 2000000000;
const ll infLL = 9000000000000000000;
```

```
#define mem(a,b) memset(a, b, sizeof(a))
#define gcd(a,b) __gcd(a,b) // GCD
ll lcm(ll a, ll b) {return (a / gcd(a, b) * b);} // LCM
#define sqr(a) ((a) * (a))

#define fast_io ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define fraction() cout.unsetf(ios::floatfield); cout.precision(10);
cout.setf(ios::fixed,ios::floatfield);
#define file() freopen("input.txt","r",stdin);freopen("output.txt","w",stdout);

typedef vector<int>::iterator vit;
typedef set<int>::iterator sit;

int dx[] = {0, 0, +1, -1};
int dy[] = {+1, -1, 0, 0};
//int dx[] = {+1, 0, -1, 0, +1, +1, -1, -1};
//int dy[] = {0, +1, 0, -1, +1, -1, +1, -1};


// Squared Distance between two point
struct Point { ll x, y; };
ll dist2(const Point &a, const Point &b) {
    ll dx = a.x - b.x;
    ll dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// String Palindrome
bool isPalindrome(const string &s) {
    ll i = 0, j = s.size() - 1;
    while (i < j) {
        if (s[i++] != s[j--]) return false;
    }
    return true;
}
// Prime Number
bool isPrime(ll n) {
    if (n <= 1) {return false;}
    for (ll i = 2; i * i <= n; i++) {
        if (n % i == 0) {return false;}
    }
    return true;
}
// Prime Generator
vector<ll> prime_gen(ll n) {
    vector<ll> primes;
    vector<bool> is_prime(n+1, true);
    is_prime[0] = is_prime[1] = false;
    for (ll i = 2; i <= n; i++) {
        if (is_prime[i]) {
            primes.push_back(i);
            for (ll j = i * i; j <= n; j += i) {
                is_prime[j] = false;
```

```
            }
        }
    }
    return primes;
}
// SemiPrime
ll semiPrime(ll a) {
     ll ct = 0;
     for (ll i = 2; ct < 2 && i * i <= a; i++) {
          while (a % i == 0) {
               a /= i;ct++;
          }
     }
     if (a > 1) ct++;
     return ct == 2;
}
// CoPrime
bool isCoprime(ll a, ll b) {
    return gcd(a, b) == 1;
}
//pair comparator
bool cmp(const pll& p1, const pll& p2) {
     if (p1.first < p2.first){
          return 1;
     }
     else if (p1.first == p2.first) {
          return(p1.second < p2.second);
     }
     return 0;
}
// Factorial
ll fact(ll n) {
    if (n == 0) { return 1; }
    ll res = ((n % mod) * (fact(n - 1) % mod)) % mod; // (n * fact(n - 1)) % mod;
    return res;
}
// Prefix Sum
//preSum[i] = preSum[i - 1] + arr[i];


template < typename F, typename S >
ostream& operator << ( ostream& os, const pair< F, S > & p ) {
    return os << "(" << p.first << ", " << p.second << ")";
}

template < typename T >
ostream &operator << ( ostream & os, const vector< T > &v ) {
    os << "{";
    for(auto it = v.begin(); it != v.end(); ++it) {
        if( it != v.begin() ) os << ", ";
        os << *it;
    }
```

```
    return os << "}";
}

template < typename T >
ostream &operator << ( ostream & os, const set< T > &v ) {
    os << "[";
    for(auto it = v.begin(); it != v.end(); ++it) {
        if( it != v.begin() ) os << ", ";
        os << *it;
    }
    return os << "]";
}

template < typename F, typename S >
ostream &operator << ( ostream & os, const map< F, S > &v ) {
    os << "[";
    for(auto it = v.begin(); it != v.end(); ++it) {
        if( it != v.begin() ) os << ", ";
        os << it -> first << " = " << it -> second ;
    }
    return os << "]";
}

#define dbg(args...) do {cerr << #args << " : "; faltu(args); } while(0)

void faltu () {
    cerr << endl;
}

template <typename T>
void faltu( T a[], int n ) {
    for(int i = 0; i < n; ++i) cerr << a[i] << ' ';
    cerr << endl;
}

template <typename T, typename ... hello>
void faltu( T arg, const hello &... rest) {
    cerr << arg << ' ';
    faltu(rest...);
}


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

1. **Coordinate Geometry**

- **Distance between two points**
  $d = \sqrt{(x_2 - x_1)^2 + (y_2 - \gamma_1)^2}$
- **Midpoint of segment**
  $M\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$
- **Slope of line**
  $m = \frac{y_2 - y_1}{x_2 - x_1}$
- **Line equations**
  - *Point-slope:* $y - y_1 = m(x - x_1)$
  - *Slope-intercept:* $y = mx + c$
  - *Two-point:* $\frac{y-y_1}{y_2-y_1} = \frac{x-x_1}{x_2-x_1}$
  - *General form:* $Ax + By + C = 0$
- **Distance from point (x_0, y_0) to line $Ax + By + C = 0$**
  $\frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$
- **Angle between two lines m_1, m_2**
  $tan\theta = \left|\frac{m_1 - m_2}{1 + m_1 m_2}\right|$

2. Circles & Conics

- **Circle, center (h, k), radius r:**
  $(x - h)^2 + (y - k)^2 = r^2$
  – Circumference: $2\pi r$
  – Area: πr^2
- **Parabola (standard):**
  $y^2 = 4ax$ or $x^2 = 4ay$
- **Ellipse (centered):**
  $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
- **Hyperbola (centered):**
  $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

3. Triangles

- **Pythagorean theorem** (right △):
  $a^2 + b^2 = c^2$
- **Area**:
  - Base–height: ½ bh
  - Heron's formula: $s = \frac{a + b + c}{2}$
- **Inradius & circumradius**:
  $r_{in} = \frac{2\Delta}{a + b + c}$, $R_{circ} = \frac{abc}{4\Delta}$ { where Δ is the triangle's area.}

4. Polygons

- **Sum of interior angles** (n-gon):
  $(n-2)\,180^0$

- **Regular n-gon**:
  – Each interior angle: $\frac{(n-2)180^0}{n}$
  – Area (side s): $\frac{ns^2}{4\tan(\pi/n)}$

5. Solid Geometry

Solid	**Volume**	**Surface Area**
Prism	V = Bh	SA = 2B + Ph (base area B, perimeter P)
Cylinder	$\pi r^2 h$	$2\pi r$ (h + r)
Cone	⅓ $\pi r^2 h$	$\pi r(r+\sqrt{r^2+h^2})$
Sphere	¼ πr^3	$4\pi r^2$

6. Miscellaneous:

- **Area of rectangle:** A = l × w
- **Area of parallelogram:** A = b × h
- **Area of trapezoid:** A= $\frac{h}{2}(b_1+b_2)$
- **Sector of circle (angle θ in radians):** A = ½ $r^2\theta$