```
#include <bits/stdc++.h>

using namespace std;

#define SYN ios_base::sync_with_stdio(0);cin.tie(0);
typedef long long int LLI;
typedef unsigned long long int ULLI;

#define debug(x)         cerr<<__LINE__<<" "<<#x<<" "<<x<<endl;
#define IMAX ((unsigned)1<<31)-1
#define eps 1e-11
#define fill(a,v) memset(a,v,sizeof (a))
#define SZ(X) ((int)X.size())
#define VI vector<int>
#define VS vector<string>
#define PB push_back
#define MSI map<string,int>
#define MSLI map<string,LLI>
#define MCI map<char,int>
#define PI acos(-1.0)
#define mk make_pair
#define pLLI pair<LLI,LLI>
#define xx first
#define yy second

#define MOD 1000000007
#define RADIANS(x)       (((1.0 * x * PI) / 180.0))
#define DEGREES(x)       (((x * 180.0) / (1.0 * PI)))


///*----------------------Graph Moves----------------*/
////const int fx[]={+1,-1,+0,+0};
////const int fy[]={+0,+0,+1,-1};
////const int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};   // Kings Move
////const int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};  // Kings Move
////const int fx[]={-2, -2, -1, -1,  1,  1,  2,  2};  // Knights Move
////const int fy[]={-1,  1, -2,  2, -2,  2, -1,  1}; // Knights Move
///*-----------------------------------------------*/



template<typename T> inline bool isOn(T mask,T pos)
{
    return mask&(1<<pos);
}

template<typename T> inline int Off(T mask,T pos)
{
    return mask^(1<<pos);
}

template<typename T> inline int On(T mask,T pos)
{
    return mask|(1<<pos);
}

template<class T> inline int countbit(T n)
{
    return (n == 0) ? 0 : (1+countbit(n&(n-1)));
}
```

```
template<class T> string toBinary(T n)
{
    string ret="";
    while(n)
    {
        if(n%2==1) ret+='1';
        else ret+='0';
        n>>=1;
    }
    reverse(ret.begin(),ret.end());
    return ret;
}


/// String Conversion template start


template<class T> string toString(T n)
{
    ostringstream ost;
    ost << n;
    ost.flush();
    return ost.str();
}

int toInt(string s)
{
    int r = 0;
    istringstream sin(s);
    sin >> r;
    return r;
}
LLI toLInt(string s)
{
    LLI r = 0;
    istringstream sin(s);
    sin >> r;
    return r;
}
double toDouble(string s)
{
    double r = 0;
    istringstream sin(s);
    sin >> r;
    return r;
}

vector <string> token( string str,string deli )
{
    char * cstr, *p,*deli_c;
    deli_c = new char [deli.size()+1];
    cstr = new char [str.size()+1];
    strcpy(deli_c,deli.c_str());
    strcpy (cstr, str.c_str());
    VS vec;
    p=strtok(cstr,deli_c);
    while(p!=NULL)
    {
        vec.push_back( string(p));
```

```
        p=strtok(NULL,deli_c);
    }
    delete[] cstr;
    return vec;
}

/// String Conversion template End


/// Math template  start
template<class T> inline T GCD(T a,T b)
{
    if ( a < 0 ) return GCD(-a,b);
    if ( b < 0 ) return GCD(a,-b);
    return (b == 0) ? a :GCD(b,a%b);
}
template<class T > inline T LCM(T a, T b)
{
    if( a < 0 ) return LCM( -a, b);
    if( b < 0 ) return LCM(a,-b);
    return a*(b/GCD(a,b));
}

template<class T> inline T BIGMOD( T n,T m,T mod )
{
    LLI ans = 1;
    LLI k = n;
    while(m)
    {
        if( m & 1)
        {
            ans *=k;
            if( ans > mod ) ans %=mod;
        }
        k *=k;
        if(k>mod) k %= mod;
        m>>=1;
    }
    return ans;
}

template < class T > T modInverse(T b, T m)
{
    return BIGMOD(b, m-2, m);
}

LLI fastExpo( LLI b, LLI p)
{
    LLI res=1,x=b;
    while(p)
    {
        if(p&1) res*=x;
        x*=x;
        p=p>>1;
    }
    return res;
}

/// Math template End
```

```
LLI hash_table( char c[] )
{
    LLI len=strlen(c);
    LLI hash_value=0,base=26,mod=10000007;

    for( LLI i=0; i<len; i++)
    {
        LLI n=c[i]-64;
        LLI power=BIGMOD(base,i,mod);
        n= (( n%mod )*( power%mod )) %mod;
        hash_value= ((hash_value%mod) +(n%mod))%mod;
    }
    return hash_value;
}

double dist(int x,int y,int x1,int y1)
{
    return sqrt(((x1-x)*(x1-x))+((y1-y)*(y1-y)));
}
```