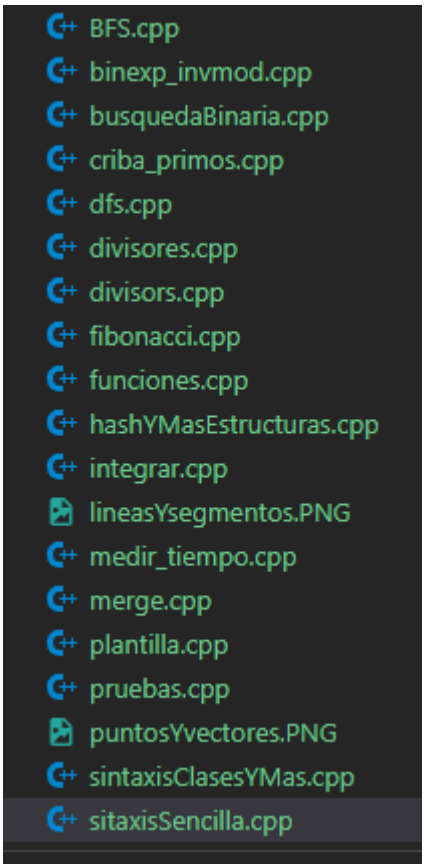
apuntes tap ashe

g++ main.cpp -o nombredelejecutable

gcc main.c -o nombredelexe




```
const int MAXN = 200005;
typedef long long tipo;
tipo INF = (tipo)(1e18+7);


struct nodo {
    tipo d; bool visto; //d -> distance, visto -> seen
};


vector<nodo> BFS(int start, int n, vector<vector<int>> &g) {
    vector<nodo> ans(n); queue<int> q;
    forn(i, n) ans[i] = {INF,false};
    ans[start] = {0,true}; q.push(start);
    while(!q.empty()) {
```

```
        int v = q.front(); q.pop();
        for(int u : g[v]) {
            if(ans[u].visto) continue;
            ans[u] = {ans[v].d+1, true}; q.push(u);
        }
    }
    return ans;
}
```

—----------

```
ll be(ll x, ll y, ll m) {
    if (y == 0) return 1;
    ll p = be(x, y/2, m) % m;
    p = (p * p) % m;
    return (y%2 == 0)? p : (x * p) % m;
}
```

```
ll be_it(ll b, ll e, ll m) {
    ll r=1; b%=m;
    while(e){if(e&1LL)r=r*b%m;b=b*b%m;e/=2;}
    return r;
}
```

```
ll inv_mod(ll x, ll m) {return be(x,m-2,m);}
```

—----------

```
1 int bsearch(const vector<int> &arr , int r ) {
2 int lef t = 0, right = n;
3 while ( right□left >1) {
4 int mid = ( lef t+right ) /2;
5 i f ( arr [mid] <= r ) {
6 lef t = mid;
7 } else {
8 right = mid;
9 }
10 }
11 return lef t ;
12 }
```

—----------

```
// Criba lineal, obtiene los primos menores al parametro
vi min_prime; // min_prime[i] contiene el menor primo que divide a i, util para factorizar en log(i)


vi criba(ll n) {
    vb prime(n+1,true);
    min_prime.resize(n+1,INF);
    vi primos;
    for(ll p=2; p*p<=n; p++){
        if(!prime[p]) continue;
        for(ll i=p*p; i<=n; i += p) {
            prime[i] = false;
            min_prime[i] = min(min_prime[i],p);
        }
    }
    forr(i, 2, n+1){
        if(prime[i]) primos.pb(i), min_prime[i] = i;
    }
    return primos; // lista de primos hasta n
}

—----------------------

const int MAXN = 200005;
vector< vector<int> > g; // graph represented as an adjacency list
vector <bool> visto(MAXN,false);


void dfs(int v) {
    visto[v] = true;
    for (int u : g[v]) if (!visto[u]) dfs(u);
}


// dfs for directed graphs
void dfs_visit(VVI& adj,int u,VI& color,VI& pred,VI& ts){
color[u] = 1;
REP(i,SIZE(adj[u])){
int v = adj[u][i];
if( color[v] == 0 ){ // (u,v): tree edge
pred[v] = u;
dfs_visit(adj,v,color,pred,ts);
}else if( color[v] == 1 ) {
 // (u,v): back-edge; cycle found (might be self-loop)
 }else{
 assert( color[v] == 2 );
```

```
// (u,v): forward or cross edge
}
}
color[u] = 2;
ts.push_back(u);
}


VI dfs(VVI& adj,VI& pred){
int n = adj.size();
pred = VI(n,-1);
VI color(n,0);
VI ts;
REP(u,n) if( color[u] == 0 ) dfs_visit(adj,u,color,pred,ts);
reverse(ALL(ts));
return ts;
}
```

—----------

```
1 i n t d [MAXD] , D=0;
2
3 void d i v ( i n t cur , i n t f [ ] , i n t s , i n t e ) {
4 i f ( s == e ) d [D++] = cur ;
5 e l s e {
6 i n t m;
7 f o r (m=s+1; m<e && f [m]==f [ s ] ; m++) ;
8 f o r ( i n t i=s ; i<=m; i++) {
9 d i v ( cur , f , m, e ) ;
10 cur = f [ s ] ;
11 }
12 }
13 }
```

—----------

```
vi find_divisors(ll n, vi &primos) {
    vector <pair<ll,ll>> factor;
    for(ll prime : primos) {
        int cont = 0;
        while(n % prime == 0) {
            cont++;
            n /= prime;
        }
        if(cont > 0) factor.pb({prime,cont});
    }

    if(n > 1) factor.pb({n,1});
```

```
    vi divisores = {1};
    for(auto [p,exp] : factor) {
        int tam = SZ(divisores);
        forn(i,exp) {
            forn(j,tam) {
                int pos = SZ(divisores)-tam;
                divisores.pb(divisores[pos] * p);
            }
        }
    }

    sort(all(divisores));
    return divisores;
}
```

—----------

```
LL fib(LL n) {
VVLL mat(2,VLL(2,1));
mat[1][1]=0;
return __gnu_cxx::power(mat,n)[1][0];
}
```

—----------

DATA STRUCTURE - C++

---------- VECTOR ----------

```
vector <ll> v; //DECLARACION DEL VECTOR, TAMANO 0.

vector <ll> v(n); //DECLARACION DEL VECTOR DE TAMANO N

v.push_back(n); //PUSHEA N AL FINAL DEL VECTOR, O(1)

sort(v.begin(),v.end()); //ORDENA EL VECTOR, O(n.logn)
```

v.pop_back(); //ELIMINA EL ULTIMO VALOR DEL VECTOR, O(1);

v.size(); //RETORNA EL TAMANO DEL VECTOR

v.clear(); //BORRA TODO EL VECTOR

v.resize(n); //ESTABLECEMOS EL TAMANO DEL VECTOR A TAMANO N

vector<pair<ll,ll>> v; //VECTOR DE PAIR

v.push_back(make_pair(a,b)); //PUSHEA EL PAR (A,B).

v[i].first; //RETORNA EL PRIMER VALOR DEL PAIR

v[i].second; //RETORNA EL SEGUNDO VALOR DEL PAIR

for(auto u : v) //RECORRE TODOS LOS VALORES DEL VECTOR

BUENO PARA STACKEAR COSAS, PARA USARLO COMO LISTA DE ADYACENCIAS
EN LOS PROBLEMAS DE GRAFOS, PARA USARLOS COMO ARREGLO.
SI SE ORDENA UN VECTOR DE PAIR, LOS ORDENA DE MENOR A MAYOR SEGUN
EL PRIMER VALOR, Y LUEGO DE MENOR A MAYOR SEGUN EL SEGUNDO.

---------- SET ----------

set <int> s; //DECLARACION DEL SET, EL SET GUARDA LOS ELEMENTOS UNA SOLA
//VEZ, Y MIENTRAS LOS VA GUARDANDO, LOS VA ORDENANDO.

multiset <int> s; //DECLARACION DEL MULTISET. EL MULTISET PUEDE GUARDAR

```
            //VARIAS COPIAS DEL MISMO NUMERO, Y MIENTRAS LOS VA
            //GUARDANDO, LOS VA ORDENANDO.


s.insert(n); //INSERTA EL VALOR N EN EL SET, SI YA ESTABA INSERTADO, NO
          //NO SE MODIFICA NADA. O(logn)


s.erase(n); //ELIMINA EL VALOR N DEL SET, O(log n)


s.size(); //TAMANO DEL SET


s.clear(); //BORRA TODOS LOS ELEMENTOS DEL SET O(s.size())


auto it=s.begin(); //DECLARACION DEL ITERATOR IT PARA EL SET


it=s.find(n); //HALLA LA POSICION DEL VALOR N DENTRO DEL SET, SI NO
           //CREO QUE RETORNA s.end(); (FINAL DEL SET)


it.lower_bound(n); //HERRAMIENTA FUERTISIMA DEL SET. CON BUSQUENA
      //BINARIA OBTIENE EL PRIMER VALOR DEL SET QUE ES MAYOR O IGUAL
      //A N, O(logn)


it=upper_bound(n); //HERRAMIENTA TAMBIEN MUY FUERTE, PERO NO LA SUELO
        //USAR MUCHO, SUELE PODER SER REEMPLAZADA POR EL LOWER_BOUND
        //EN O(logn) DEVUELVE EL PRIMER VALOR MAYOR ESTRICTO QUE
        //N


for(auto it : s) //RECORRE TODOS LOS VALORES DEL SET


// Custom comparators
struct Edge {
   int a, b, w;
};
struct cmp { // Funcion comparadora para el set
   bool operator()(const Edge &x, const Edge &y) const { return x.w < y.w; }
};


int main() {
```

```
    int M = 4;
    set<Edge, cmp> v;
    ...


// FACIL DE USAR. PUEDE REEMPLAZAR EL PRIORITY QUEUE PERO ES UN POQUITO
MAS
// LENTO. BUENO PARA PROBLEMA QUE CONVIENE IR ORDENANDO EN TIEMPO
REAL,
// EN VEZ DE LEER TODO Y LUEGO ORDENARLO. SE USA MUCHO EN ALGORITMOS
PARA
// GRAFOS COMO DIJKSTRA Y PRIM.




------------------------------------------------------------


---------- QUEUE ------------------------------


queue <int> q; //DECLARACION DEL QUEUE


q.push(n); //PUSHEA AL FINAL DEL QUEUE EL NUMERO N


q.front(); //OBTIENE EL NUMERO QUE SE ENCUENTRA PRIMERO EN LA QUEUE


q.pop() //BORRA EL NUMERO QUE SE ENCUENTRA PRIMERO EN EL QUEUE


------------------------------------------------------------


---------- PRIORITY QUEUE ----------


prority_queue <int> pq; //DECLARACION DEL PRIORITY_QUEUE. A MEDIDA QUE
                        //VAMOS INSERTANDO VALORES, LOS VA ORDENANDO EN
                        //ORDEN CRECIENTE.


priority_queue<int, vector<int>, greater<int>> pq; // PRIORITY_QUEUE DE MINIMO
```

pq.push(n); //INSERTA EL VALOR N EN EL LUGAR QUE CORRESPONDE. O(log n)

pq.top(); //OBTIENE EL VALOR QUE SE ENCUENTRA PRIMERO EN LA PQ.

pq.pop(); //ELIMINA EL ELEMENTO QUE SE ENCUENTRA PRIMERO EN LA PQ.

—----------

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


struct hash_string {
    hash<char*> h;
    size_t operator()(const string &s) const { return h(s.c_str()); }
};


void hash_demo() {
    hash_map<string, string, hash_string> foo;
    foo["foo"] = "bar";
    cout << foo["foo"] << endl;
}


int main() {
    // Pila (Stack)
    stack<int> pila;
    pila.push(10);
    pila.push(20);
    pila.push(30);
    cout << "Top de la pila: " << pila.top() << endl;
    pila.pop();
    cout << "Top de la pila después de pop: " << pila.top() << endl;


    // Lista (List)
    list<int> lista;
    lista.push_back(10);
    lista.push_back(20);
    lista.push_back(30);
    cout << "Elementos de la lista: ";
    for(int elem : lista) {
        cout << elem << " ";
```

```
    }
    cout << endl;


    // Hash (unordered_map)
    unordered_map<string, int> hash_table;
    hash_table["uno"] = 1;
    hash_table["dos"] = 2;
    hash_table["tres"] = 3;
    cout << "Valor asociado a 'dos': " << hash_table["dos"] << endl;


    // Conjunto (Set)
    set<int> conjunto;
    conjunto.insert(10);
    conjunto.insert(20);
    conjunto.insert(30);
    cout << "Elementos del conjunto: ";
    for(int elem : conjunto) {
        cout << elem << " ";
    }
    cout << endl;


    // Diccionario (unordered_map, similar a hash)
    unordered_map<string, int> diccionario;
    diccionario["uno"] = 1;
    diccionario["dos"] = 2;
    diccionario["tres"] = 3;
    cout << "Valor asociado a 'tres' en diccionario: " << diccionario["tres"] << endl;


    return 0;
}



—----------------------------------

static double simpson2(double a, double b, double del, int step, int maxstep) {
double h = b-a;
double c = (b+a)/2;
double fa = f(a);
double fc = f(c);
double fb = f(b);
double s0 = h*(fa+4*fc+fb)/6;
double s1 = h*(fa+4*f(a+h/4)+2*fc + 4*f(a+3*h/4)+fb)/12;
step++;
```

```
if (step >= maxstep) {
System.out.println ("Not converged after " + step + " recursions");
return s1;
} else {
if (Math.abs(s1-s0) < 15*del)
return s1;
else
return simpson2(a, c, del/2, step, maxstep) +
simpson2(c, b, del/2, step, maxstep);
}
}
static double monte(int steps) {
Random r = new Random(); double s0 = 0; double ds = 0;
for (int i=0; i<steps; ++i) {
double x = r.nextDouble();
s0 += f(x); ds += f(x)*f(x);
}
s0 /= steps; ds /= steps;
ds = Math.sqrt(Math.abs(ds-s0*s0)/steps); // error
return s0;
}
static double f(double x) { ... }
```

—----------

```
struct line {
    double a, b, c;
    line(double aa=0.0, double bb=0.0, double cc=0.0) {
        a=aa; b=bb; c=cc;
    }
};

double dist(const pt &p, const line &l) {
    return ABS(l.a*p.x+l.b*p.y+l.c)/sqrt(SQ(l.a)+SQ(l.b)); }

line line_pp(const pt &p1, const pt &p2) {
    return line(p2.y-p1.y, p1.x-p2.x, p2^p1); }

line line_perp_p(const line &l, const pt &p) {
    return line(-l.b, l.a, l.b*p.x - l.a*p.y); }

line mediatriz(const pt &p1, const pt &p2) {
    return line_perp_p(line_pp(p1, p2), (p1+p2)/2.0); }
```

—----------

```
unsigned t0, t1;
t0=clock();
```

```
// Aca Va el codigo

t1 = clock();
double time = (double(t1-t0)/CLOCKS_PER_SEC);
cout << "Execution Time: " << time << endl;



—----------------------------

// cnt_inv = number of inversions in A
LL cnt_inv = 0;


void merge(VLL& A,int a,int b,int c){
VLL B(A.begin()+a, A.begin()+b);
VLL C(A.begin()+b, A.begin()+c);
int i=0,j=0,k=a;
while(i<B.size() && j<C.size()){
if(B[i] <= C[j]){
 A[k++] = B[i++];
 }else{
 cnt_inv += B.size()-i;
 A[k++] = C[j++];
 }
 }
 while(i<B.size()) A[k++] = B[i++];
 while(j<C.size()) A[k++] = C[j++];
 }
 void mergesort(VLL& A,int l,int u){
 if(u-l > 1){
 int m = (l+u) / 2;
 mergesort(A,l,m);
 mergesort(A,m,u);
 merge(A,l,m,u);
 }
 }

—----------------------------

/*   AUTHOR: Estufa en Piloto   */
#include <bits/stdc++.h>
using namespace std;


#ifdef LOCAL
#define DBG(x) cerr << #x << " = " << (x) << endl
```

```
#define RAYA cerr << "==============================" << endl
#else
#define DBG(x)
#define RAYA
#endif


typedef long long ll;
typedef vector<ll> vi; typedef pair<ll,ll> ii;
typedef vector<ii> vii; typedef vector<bool> vb;
#define FIN ios::sync_with_stdio(0);cin.tie(0);cout.tie(0)
#define forr(i, a, b) for(ll i = (a); i < (ll) (b); i++)
#define forn(i, n) forr(i, 0, n)
#define SZ(x) int((x).size())
#define pb push_back
#define mp make_pair
#define all(c) (c).begin(),(c).end()
#define esta(x,c) ((c).find(x) != (c).end())
const int INF = 1<<30; // const ll INF = (1LL<<60);
const int MOD = 1e9+7;  // const int MOD = 998244353;
const int MAXN  = 2e5+5;


int main(){
    FIN;







    return 0;
}



—----------------------------------

#include <iostream>
#include <cmath>


using namespace std;


int sumaDivisores(int n);
```

```
int main() {
    int i;
    cin >> i;
    int n;
    for (int j = 0; j < i; j++) {

        cin >> n;
        int suma = sumaDivisores(n);
        cout << n;


        if (suma == n) {
            cout << " perfecto";
        } else {
            int sumaDivSuma = sumaDivisores(suma);
            if (n == sumaDivSuma) {
                cout << " romantico";
            }
            if (n < suma) {
                cout << " abundante";
            }
            if (n != sumaDivSuma && n >= suma) {
                cout << " complicado";
            }
        }
        cout << "\n";
    }
    return 0;
}


int sumaDivisores(int n) {
    if (n == 1) return 0; // El número 1 no tiene divisores propios


    int suma = 0; // Iniciamos en 0, ya que 1 siempre es un divisor propio
    int raiz = sqrt(n);


    for (int i = 1; i <= raiz; i++) {
        if (n % i == 0) {
            if (i != n) suma += i; // Añadimos i si no es el número mismo
            if (i != 1 && i != n / i && n / i != n) suma += n / i; // Añadimos n / i si no es el número
mismo ni el divisor ya añadido
        }
    }
```

```
    return suma;
}
```

—----------

```
struct pt {
    double x, y;
    pt(double xx=0.0, double yy=0.0) { x=xx; y=yy; }
};

pt operator+(const pt &p1, const pt &p2) {
    return pt(p1.x+p2.x, p1.y+p2.y); }

pt operator-(const pt &p1, const pt &p2) {
    return pt(p1.x-p2.x, p1.y-p2.y); }

double operator*(const pt &p1, const pt &p2) {
    return p1.x*p2.x + p1.y*p2.y; }

double operator^(const pt &p1, const pt &p2) {
    return p1.x*p2.y - p1.y*p2.x; }

double norm(const pt &p) { return sqrt(p*p); }

double dist(const pt &p1, const pt &p2) {
    return norm(p1-p2); }
```

—----------

```
#include <iostream>
#include <string>
using namespace std;
```

```
class Padre {
  private:
    string name;
};


class OtroPadre {
  private:
    string lastname;
};


class MyClass : public Padre, public OtroPadre{ //extiende de padre y otropadre
    private:
        int myNum;
        string myString;


    protected:
        string otroString;


    public:
        MyClass() {     // Constructor
            myNum = 0;
            myString = "";
        }


        void setNum(int num) {
            myNum = num;
        }


        int getNum(){
            return myNum;
        }


};


int main(){
    MyClass myObj;
    myObj.setNum(15);
    cout << "myNum: " << myObj.getNum() <<  "\n";
```

```
    //excepciones
    try {
    int age = 15;
    if (age >= 18) {
        cout << "Access granted - you are old enough.";
    } else {
        throw (age);
    }
    }
    catch (int myNum) {
    cout << "Access denied - You must be at least 18 years old.\n";
    cout << "Age is: " << myNum;
    }


    //random
    srand(time(NULL));//srand(time(nullptr))
    int x=rand();
    //rand()) / RAND_MAX * positionRange - positionRange / 2


    return 0;
}


—----------------------------------


#include <iostream>
#include <string> //para poder usar string
#include <cmath>
using namespace std; //evita tener que poner std::


//this is a comment
/*this is ohter comment*/


int myFunction(string fname, int age) { //paso por valor
  cout << fname << " Refsnes. " << age << " years old. \n";
  return age;
}


void swapNums(int &x, int &y) {//paso por referencia -> se modifican los parámetros reales
  int z = x;
```

```
  x = y;
  y = z;
}

int main() {
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);//mejora la eficiencia


    //primitivos
    const int CONSTANTE = 5; //si o si le debemos definir el valor ahora
    int entero1, entero2, entero3;
    double real = 0.0;
    float real2 = 0.0; //menos preciso y MAS RAPIDO que double (32 bits en lugar de 64)
    char caracter = 'a';
    bool boleano = false; //en pantalla imprime un 0, no "false"


    //string
    string texto = "hola que tal";
    char greeting2[] = "Hello";  //     MENOS EFICIENTE
    string nombre;
    cout << "Type your name: ";
    getline (cin, nombre);
    string segundoNombre = "Agustin ";
    string apellido = "Ibanez ";
    string fullName = nombre.append(segundoNombre) + apellido; //dos formas de
concatenar strings
    cout << fullName <<  "\n";
    cout << "The length of the fullname is: " << fullName.length() << "\n"; //EL \N ES MÁS
RÁPIDO QUE ENDL
    cout << "La primer letra del nombre es " << fullName[0] << "\n";


    //imprimir
    cout << "Type a number: " <<  "\n"; //esta es la forma más rápida de hacer un salto de
línea
    cin >> entero1; //el input usa << y el uotput >>


    // condicionales
    if((entero1 != 50)&&(entero1 !=49)){
        cout << "Entero1 no es 50 ni 49";
    }
    else{
        if(entero2 >50 && entero2<100){
            cout << "Entero 2 entre 50 y 100";
        }
        else if(entero3 <23){
```

```
        cout << "entero3 menor a 23";
    }
}


int time = 20;
string result = (time < 18) ? "Good day." : "Good evening.";
cout << result;


int day = 4;
switch (day) {
case 6:
    cout << "Today is Saturday";
    break;
case 7:
    cout << "Today is Sunday";
    break;
default:
    cout << "Looking forward to the Weekend";
}


//matemática
cout << sqrt(64);
cout << round(2.6);
cout << log(2);
cout << min(5, 10);
return 0;


//iteradores
int i = 0;
while (i < 5) {
    cout << i << "\n";
    i++;
}


int i = 0;
do {
    cout << i << "\n";
    i++;
}
while (i < 5);

for (int i = 0; i < 5; i++) {
    cout << i << "\n";
```

```
    }


    int myNumbers[5] = {10, 20, 30, 40, 50};
    for (int i : myNumbers) {

        if ( i == 20 ){
            continue; //saltamos a la siguiente iteración del for
        }
        else if ( i == 40 ){
            break; //salimos del for, nunca llegamos al 50
        }
    }


    //arreglos
    int myNumbers[5] = {10, 20, 30, 40, 50};
    cout << sizeof(myNumbers); //tamaño en bytes
    int getArrayLength = sizeof(myNumbers) / sizeof(int); //cantidad de elementos
    cout << getArrayLength;
    int otherNumbers[10] = {0}; //si o si hay que aclarar el tamaño del arreglo, de esta forma
inicia todo en 0


    string letters[2][4] = {
        { "A", "B", "C", "D" },
        { "E", "F", "G", "H" }
    };
    cout << letters[0][2];  // Outputs "C"


    //estructuras
    struct tipoEstructura{
        int myNum;
        string myString;
    };


    //punteros
    string food = "Pizza";  // A food variable of type string
    string* ptr = &food;    //con & pido la dirección de una variable y con * indico que es un tipo
puntero y tmb me sirve para referenciar al valor al que apunta
    cout << *ptr << "\n"; //muestro el valor que apunta
    cout << ptr << "\n"; //muestro la dirección a la que apunta
    cout << &ptr << "\n"; //muestro al dirección donde guardo la variable
}
```