../10107-uva-busca-binaria.cpp

```cpp
//qual a mediana? UVA 10107

#include <bits/stdc++.h>
#define F first
#define S second
#define mp make_pair
#define pb push_back //insere sempre no final
#define INF 0x3f3f3f3f //infinito
#define LINF 0x3f3f3f3f3f3f3f3fLL
using namespace std;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> ii;
typedef vector<ii> vii;

vector <int> v;

int main(){
  // insert desloca todo mundo para a esquerda e adiciona o item a esquerda
  int x;
  while(scanf("%d", &x)!= EOF){
    vector<int>::iterator it; // tem que ser a estrutura que deseja ser percorrida
    for(it = v.begin(); it != v.end(); it++){ //end, depois do ultimo
      if(*it >= x)
        break;
    }

    v.insert(it,x); //insere ordenado (posicao (tem q ser um iterator), valor)
    int mid = v.size()/2;
    int ans;
    if(v.size()%2== 0){ // tamanho par
      ans = (v[mid] + v[mid-1])/2;
    }else{
      ans = v[mid];
    }
    printf("%d\n", ans);
  }

}
```

../10954-uva.cpp

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

using namespace std;

//UVA10954
```

```cpp
int main(){
  int n;

  while(scanf("%d",&n) && n)  {
    priority_queue<int,vector<int>,greater<int> > pq; //vector<int>, greater<int> (base(sempre vector do tipo)/comparador)
    int x;                          //(necessario apenas se especificar comparador)
    for(int i=0;i<n;i++)    {
      scanf("%d",&x);
      pq.push(x);
    }
    int total=0;
    while(pq.size()>1)    {
      int a=pq.top();
      pq.pop();
      int b=pq.top();
      pq.pop();
      pq.push(a+b);
      total+=a+b;
    }
    printf("%d\n",total);
  }

  return 0;
}
```

../11034-uva.cpp

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

int main(){
  int c;
  scanf("%d", &c);
  while(c--){
    int l, m, a, cont = 0;
    string side;
    queue<int> left, right;
    scanf("%d %d", &l, &m);
    l *= 100; // m para cm

    for(int i=0; i < m; i++){
      cin >> a >> side;
      if(side == "left") left.push(a);
      else right.push(a);
    }

    while(!left.empty() || !right.empty()){
      int load = 0;
      while(!left.empty() && load + left.front() <= l){
        load += left.front();
```

```
24          left.pop();
25        }
26        if(load == 0 && right.empty()) break;
27        cont++;
28        load = 0;
29        while(!right.empty() && load + right.front() <= l){
30          load += right.front();
31          right.pop();
32        }
33        if(load == 0 && left.empty()) break;
34        cont++;
35      }
36      printf("%d\n", cont);
37    }
38    return 0;
39  }
```

../11849-uva-set.cpp

```
1   //cd
2   //todo elemento no set    um valor  nico
3   #include <bits/stdc++.h>
4   #define pb push_back //insere sempre no final
5   using namespace std;
6
7   int main(){
8     set<int> jack, jill;
9     int n,m;
10    while(scanf("%d %d", &n, &m)!=EOF && (n | m)){ // ou bitwase == if
11      for (int i = 0; i < n; ++i) {
12        int x;
13        scanf("%d", &x);
14        jack.insert(x);
15      }
16      for (int i = 0; i < m; ++i) {
17        int x;
18        scanf("%d", &x);
19        jill.insert(x);
20      }
21      set<int>::iterator it;
22      int cont = 0;
23      for(it = jack.begin(); it!= jack.end(); ++it){ // percorre a arvore em
            ordem (como um vertor ordenado)
24        int x = *it; // menor valor da arvore == mais a esquerda possivel
25        if(jill.find(x) != jill.end()) { // retorna um iterator para a
              posi  o do elemento se ele existir, se nao existir, retorna o
              end do set
26          ++cont;
27        }
28      }
29      printf("%d\n", cont);
30      jack.clear();
```

```
    jill.clear();
  }
  return 0;
}
```

../11849-uva-vector.cpp

```
//cd
//todo elemento no set    um valor  nico
#include <bits/stdc++.h>
#define pb push_back //insere sempre no final
using namespace std;

int main(){
  vector<int> jack;
  set<int> jill;
  int n,m;
  while(scanf("%d %d", &n, &m)!=EOF && (n | m)){ // ou bitwase == if
    for (int i = 0; i < n; ++i) {
      int x;
      scanf("%d", &x);
      jack.pb(x);
    }
    for (int i = 0; i < m; ++i) {
      int x;
      scanf("%d", &x);
      jill.insert(x);
    }
    int cont = 0;
    for(int i= 0; i < jack.size(); ++i){ // percorre a arvore em ordem (
        como um vertor ordenado)
      if(jill.count(jack[i])) { //sempre retorna ou 1 ou 0
        ++cont;
      }
    }
    printf("%d\n", cont);
  }
  return 0;
}
```

../1260-uri-map.cpp

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

int main(){ // map, chaves unicas (ordenado)
  int n;
  bool flag = false;
  scanf("%d", &n);
  getchar(); // limpa o buffer
```

```
9    getchar(); // tira a primeira linha em branco (separa os casos)
10
11   while(n--){ // primeiro roda o loop, depois decrementa
12     if(flag) printf("\n");
13     flag = true;
14
15     map<string, int> ars;
16     string arv;
17     int count = 0;
18     while(getline(cin, arv) && !arv.empty()){ //leitura ate encontrar uma
           linha. leitura de onde? entrada padrao (cin), e salva na string .
           getline l  somente at  o enter e o tira do buffer
19       ars[arv]++; //todo elemento    inicializado com zero no map, j
             cria e ja conta
20       count++; // conta a quantidade de arvores
21     }
22     map<string, int>::iterator it;
23     for(it = ars.begin(); it != ars.end(); it++){ //percorre de forma
           ordenada
24       printf("%s %.4lf\n", it->first.c_str(), 100.0*it->second/count);//
             === (*it).first (*it).second
25     }
26   }
27
28   return 0;
29 }
```

../1610-uri.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5
6  }
```

../514-uva.cpp

```
1  //pilha -trilhos no URI
2  //FILO - LIFO
3  //
4  // estacao    uma pilha
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  int main(){
9    int n;
10   while(scanf("%d", &n) && n){
11     int x;
12     while(scanf("%d", &x) && x){
```

```
13        queue<int> q;
14        stack<int> s;
15        q.push(x);
16
17        for(int i=1; i < n; i++){
18          scanf("%d", &x);
19          q.push(x);
20        }
21
22        x = 1; // vagao que esta entrando
23        while(!q.empty()){
24          if(x == n + 2) break; // todos os vagoes ja entraram
25          while(!s.empty() && q.front() == s.top()){
26            q.pop();
27            s.pop();
28          }
29
30          if(q.empty()) break;
31
32          if(s.empty() || q.front() != s.top()){ // novo vagao pode entrar
33            s.push(x);
34            x++;
35          }
36        }
37        printf("%s\n", q.empty() ? "Yes" : "No");
38      }
39      printf("\n");
40    }
41
42    return 0;
43  }
```

../a.cpp

```
1   string a, b;
2
3   bool simula(int k){
4     string aux;
5     for(int i = 0; i < b.size(); i++){
6       for(int j = 0; j < k; j++){
7         aux.push_back(b[i]);
8       }
9     }
10    int cont = 0;
11    for(int i = 0; i < a.size() i++){
12      if(a[i] == aux[cont]){
13        cont++;
14      }
15    }
16
17    return cont == aux.size();
18  }
```

```
19
20 int main(){
21   while(n--){
22     cin >> a >> b;
23     int lo = 0, hi = a.size() / b.size();
24     int res = 0;
25     while(lo <= hi){
26       int mid = (lo+hi)/2;
27       if(simula(mid)){
28         lo = mid + 1;
29         res = mid;
30       }else{
31         hi = mid - 1;
32       }
33     }
34
35     printf("%d\n", res);
36
37   }
38 }
```

../a-exponenciacao-rapida.cpp

```
1  #include <bits/stdc++.h>
2  #define F first
3  #define S second
4  #define mp make_pair
5  #define pb push_back
6  #define INF 0x3f3f3f3f //infinito
7  #define LINF 0x3f3f3f3f3f3f3f3fLL
8  using namespace std;
9  typedef long long ll;
10 typedef vector<int> vi;
11 typedef pair<int, int> ii;
12 typedef vector<ii> vii;
13
14 ll mod = (1LL << 31) - 1LL;
15
16 ll fast_exp(ll e){
17     if(!e) return 1;
18     ll a = fast_exp(e/2);
19     a = (a*a) % mod;
20     if(e&1){ // operador bitwase 1, compara somente o primeiro bit ativo
21       a = (a * 3) % mod;
22     }
23     return a;
24 }
25
26 int main(){
27   int n;
28   scanf("%d", &n);
29   printf("%lld\n", fast_exp(n));
```

```
30   return 0;
31 }
```

../arranjo-iterativo.cpp

```
1  /* Uma maneira de gerar as amostras iterativamente    come ar com a
       permuta  o {1,1,..,1}. A pr xima permuta  o    gerada escaneando a
        permuta  o atual da direita para esquerda at  encontrar um elemento
        que n  o chegou no valor m ximo. Este elemento    incrementado em 1 e
        todos os elementos para a direita s  o resetados para o valor m nimo
       (1). O processo se repete at  que todas as amostras tenham sido
       geradas.
2  Escreva este algoritmo e implemente-o.
3  * */
4
5  #include <iostream>
6  #include <vector>
7
8  using namespace std;
9
10 void arranjo(int, vector<int> &);
11
12 int main(){
13
14   int          k, n;    cin >> n >> k;
15   //k -> tamanho da amostra
16   vector<int> a(k,1);
17
18   arranjo(n,a); // se jogar o k direto  j   extrapolou
19
20   return 0;
21 }
22
23 void arranjo(int n, vector<int> &a){
24   int i = a.size()-1;
25   for(i;i>=0;i--){ //percorre de tras para frente
26
27     if(a[i]<n){
28      for(auto j: a)
29       cout << j << ' ';
30      cout << endl;
31       a[i]++;
32       for(i=i+1;i<a.size();i++)
33         a[i]=1;
34
35     }
36   }
37
38   for(auto j: a)
39       cout << j << ' ';
40   cout << endl;
41
```

```
42
43 }
```

../arranjo-recursivo.cpp

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  void arranjo(int, int, vector<int> &);
7
8  int main(){
9    vector<int> a;
10   int         k, n;   cin >> n >> k;
11
12   a.resize(k);
13   arranjo(n,0,a); // se jogar o k direto  j   extrapolou
14
15   return 0;
16 }
17
18 void arranjo(int n, int k, vector<int> &a){
19   if(k==a.size()){ //indice zero
20     for(auto i: a)
21       cout << i << ' ' ;
22     cout << endl;
23     return;
24   }
25   else{
26     for(int i=1;i<=n;i++){
27       a[k]=i;
28       arranjo(n,k+1,a);
29     }
30   }
31
32 }
```

../arranjo-reverso.cpp

```
1  #include <iostream>
2  #include <vector>
3  /*
4  Implemente um algoritmo que gera as amostras em ordem reversa
5  */
6  using namespace std;
7
8  void arranjo_reverso(int, int, vector<int> &);
9
10 int main(){
```

```
  vector<int> a;
  int         k, n;   cin >> n >> k;

  a.resize(k);
  arranjo_reverso(n,0,a); // se jogar o k direto j  extrapolou

  return 0;
}

void arranjo_reverso(int n, int k, vector<int> &a){
  if((unsigned)k==a.size()){ //indice zero
    for(auto i: a)
      cout << i << ' ' ;
    cout << endl;
    return;
  }
  else{
    for(int i=n;i>=1;i--){
      a[k]=i;
      arranjo_reverso(n,k+1,a);
    }
  }

}
```

../b.cpp

```
#include <bits/stdc++.h>
#define F first
#define S second
#define mp make_pair
#define pb push_back
#define INF 0x3f3f3f3f //infinito
#define LINF 0x3f3f3f3f3f3f3f3fLL
using namespace std;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> ii;
typedef vector<ii> vii;

ii mod[] = {ii(-1, -2), // todo vertice tem oito arestas
                        ii(-1,  2),
                        ii(1 , -2),
                        ii(1 ,  2),
                        ii(-2, -1),
                        ii(-2,  1),
                        ii(2 , -1),
                        ii(2 ,  1)
                       }

int dis[10][10]; // i e j

```

```
26 bool valida(ii coord){
27   return !(coord.F < 1 || coord.F > 8 || coord.S < 1 || coord.S > 8);
28 }
29
30 void bfs(int i, int j){
31   memset(dist, INF, sizeof(dist)); // distancia para todo mundo
        infinito
32   queue<ii> q; // linha, coluna | visitar todos adjacentes | coordenadas
        de onde estou | quem posso enxergar
33
34   dist[i][j] = 0;
35   q.push(ii(i,j));
36   while(!q.empty()){
37     ii origem = q.front();
38     q.pop();
39     for(int i = 0; i < 8; i++){
40       ii destino(origem.F + mod[i].F, origem.S + mod[i].S);
41       if(valida(destino)){ // valida os limites
42         if(dist[destino.F][destino.S] == INF){ // n o foi visitado
43           q.push(destino);
44           dist[destino.F][destino.S] = dist[origem.F][origem.S] + 1; //
               custo para chegar la    o
45         }
46       }
47     }
48   }
49
50
51 }
52
53 int main(){
54   int iinicial, ifinal, jinicial, jfinal; // i - linha | j - coluna
55   char a, b;
56   while(scanf(" %c%d %c%d", &a, &iinicial, &b, &ifinal)!= EOF){
57     jinicial = a - 'a' + 1; // matriz indexada de 1
58     jfinal   = b - 'a' + 1;
59     bfs(iinicial, jinicial);
60     printf("%d\n", dist[ifinal][jfinal]);
61   }
62 }
```

../c.cpp

```
1 // KRUSKAL - MST - minimum spanning tree - arvore geradora minima - grafo
    conexo, sem ciclos, de menor custo total - nao necessariamente caminho
    minimos
2 #include <bits/stdc++.h>
3 #define F first
4 #define S second
5 #define mp make_pair
6 #define pb push_back
7 #define INF 0x3f3f3f3f //infinito
```

```cpp
#define LINF 0x3f3f3f3f3f3f3f3fLL
using namespace std;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> ii;
typedef vector<ii> vii;

class UnionFind{ // so serve para grafo nao direcionado
public:
  vi p, rank;
  UnionFind(int n){
    p.resize(n);
    rank.assign(n, 0);
    for(int i = 0; i < n; i++) p[i] = i; // pai dele    ele mesmo
  }

  int findSet(int i){
    return (p[i] == i)? i : (p[i] = findSet(p[i]));
  }

  bool isSameSet(int i, int j){ // fala se vai ter ciclo ou nao
    return findSet(i) == findSet(j);
  }

  void unionSet(int i, int j){
    int x = findSet(i);
    int y = findSet(j);
    if(x == y) return;
    if(rank[x] > rank[y]){
      p[y] = x;
      return;
    }

    p[x] = y;
    if(rank[x] == rank[y]) rank[y]++;
  }

};

class Aresta{
public:
  int peso, origem, destino;
  Aresta(int p, int o, int d){
    peso    = p;
    origem  = o;
    destino = d;
  }

  bool operator<(const Aresta a) const{
    if(peso != a.peso) return peso < a.peso;
    if(origem != a.origem) return origem < a.origem;
    return destino < a.destino;
  }
};
```

```
// set<pair<int, ii>> edgesList;
set<Aresta> edgesList;
int n, m; // quantidade de vertices

int kruskal(){ // MST
  set<Aresta>:: iterator it;
  UnionFind uf(n);
  int sum = 0;
  for(it = edgesList.begin(); it != edgesList.end(); it++){
    if(uf.isSameSet(it->origem, it->destino)) continue;
    uf.unionSet(it->origem, it->destino);
    sum += it->peso;
  }
}


int main(){
  // ordena as arestas por peso
  // perguntando se vai dar ciclo ou nao -> sabendo se o representante
     o mesmo -> liga arestas n o conectadas
  while(scanf("%d %d", &n,&m) && (n|m)){
    edgesList.clear();
    int a, b, w, cont = 0;
    for(int i = 0; i < m ; i++){
      scanf("%d %d %d", &a, &b, &w);
      Aresta j(w,a,b);
      edgesList.insert(j);
      cont += w;
    }
    printf("%d\n", cont);
  }
  return 0;
}
```

../c-map.cpp

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

int main(){
  int n;
  while(scanf("%d", &n) && n){
    map<set<int>,int> m;
    map<set<int>,int>::iterator it;
    int maior = 0, cont = 0;
    for(int i=0;i < n; i++){
      set<int> s; // grade
      for(int j=0; j < 5; j++){
        int x;
        scanf("%d", &x);
        s.insert(x);
      }
```

```
17        m[s]++;  //aumenta a frequencia da frequencia
18        if(m[s] > maior) maior = m[s]; // pega a maior frequencia
19      }
20
21      for(it = m.begin(); it != m.end(); it++){
22        if(it->second == maior)
23          cont++; // contando no caso de empate
24      }
25
26      printf("%d\n", maior*cont );
27    }
28  }
```

../contagem.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      // contagem de bits
6      long long int n;
7      // int c=0;
8      while(cin >> n, n){
9          // ------------- uma forma
10         for(int i=0;i <=32; i++){
11             if(n&(1<<i)) //shifita e faz and
12                 c++;
13         }
14         cout << c << endl;
15         //
16         // ------------- outra forma
17         cout << __builtin_popcount(n) << endl;
18         //
19         // ------------- bitset -> tranforma numero em vetor
20         bitset<40> vetor(n); //numero maximo de bits, numero que vai
21         //ser transformado
22         cout << vetor.count() << endl ;
23     }
24 }
```

../conta-qnt-digitos.cpp

```
1  #include <iostream>
2  using namespace std;
3
4
5  int contaDigitos(int);
6  int main(){
7    cout << contaDigitos(59785) << endl;
8  }
```

```
9
10 int contaDigitos(int n){
11   int conta=0;
12   do{
13     conta++;
14     n/=10;
15   }while(n!=0);
16   return conta;
17 }
```

../cosseno.cpp

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  float cosseno(float);
7
8
9  int main(){
10   float x;
11   cin >> x;
12   cout << cosseno(x) << endl;
13   return 0;
14 }
15
16 float cosseno(float x){
17 //calculo em radianos
18   int i=2, sinal=-1;
19   float t = ((x*x)/i)*sinal; //primeiro termo    negativo
20   float cos=1; //primeiro numero da sequencia    1
21   while(abs(t) > 0.0000001){
22     cos +=t; //na primeira passada soma com o negativo
23     sinal *= -1; //inverte o sinal
24     i+=2; //na primeira passada vai para 4
25     t=(t*x*x)/((i-1)*i)*sinal;
26   }
27   return cos;
28 }
```

../crivo-de-erastosteles.cpp

```
1 // descoberta de primos de numero eficiente
2 // faz preprocessamento - identifica todos os primos at  10
3 // se for divisivel por numero composto, basta dividir por primo
4 // bitset onde se o valor for zero o numero    primo, caso contrario    um
      (POSICAO)
5 // marca todos os multiplos de primo como nao primo, exceto os pares ( A
     PARTIR do quadrado dele! pq ele antes dele vai ser composto)
```

```cpp
// limite do crivo    10
// consegue achar ate o quadrado do crivo

#include <bits/stdc++.h>
#define F first
#define S second
#define mp make_pair
#define pb push_back
#define INF 0x3f3f3f3f //infinito
#define LINF 0x3f3f3f3f3f3f3f3fLL
using namespace std;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> ii;
typedef vector<ii> vii;

bitset<10000010> primo; //primo ou nao
vector<int> primos; //lista dos primos

void crivo(){
  primos.pb(2);
  primo[1] = 1;
  for(ll i = 3; i <= 10000000; i += 2){
    if(primo[i]) continue;
    for(ll j = i * i; j <= 1000000; j += i){ // come a do quadrado
      primo[j] = 1;
    }
    primos.pb(i);
  }
}

bool isPrime(ll num){
  if(num == 2) return true;
  if(num % 2 == 0) return false;
  if(num <= 10000000) return !primo[num];
  for(int i = 0; i < primos.size(); i++){
    if(num % primos[i] == 0) return false;
    if(primos[i] > sqrt(num)) break; // nao precisa ir acima da raiz
        quadrada
  }
  return true;
}


int main(){
  crivo();
  long long n;
  while(scanf("%lld", &n) && n){
    printf("%s\n", isPrime(n)? "yes": "no");
  }

}
```

../c-unionFind.cpp

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

class Union{
  public:
    vector <int> p, rank, size;

    Union(int n){
      p.resize(n);
      rank.assign(n,0);
      size.assign(n, 1);
      for(int i = 0;i<n;i++){
        p[i] = i;
      }
    }

    int UnionFind(int x){
      if(p[x] == x) return x;
      return (p[x] = UnionFind(p[x]));
    }

    bool isSame(int x, int y){
      return(UnionFind(x) == UnionFind(y));
    }

    void Unionset(int i, int j){
      if(isSame(i, j)){
        return;
      }

      int x = UnionFind(i);
      int y = UnionFind(j);

      if(rank[x] > rank[y]){
        p[y] = x;
        size[x]+=size[y]; // tamanho de x    o tamanho de y mais o dele
        return;
      }

      p[x] = y;
      size[y]+=size[x]; // tamanho de y    o tamanho de x mais o dele

      if(rank[x] == rank[y]) rank[y]++;

    }
    int sizeset(int x){
      return size[UnionFind(x)];
    }

};

int main(){
```

```
53    int a, b;
54    cin >> a;
55    while(a--){
56      map<string, int> mapa;
57      string s1, s2;
58      int cont = 0;
59      Union u(1000000);
60      cin >> b;
61      for(int i = 0;i<b;i++){
62        cin >> s1 >> s2;
63        if(mapa.count(s1) == 0){
64          mapa[s1] = cont;
65          cont++;
66        }
67        if(mapa.count(s2) == 0){
68          mapa[s2] = cont;
69          cont++;
70        }
71
72        int maior;
73        //cout << mapa[s1]  << " " << mapa[s2] << endl;
74        u.Unionset(mapa[s1], mapa[s2]);
75        int r = u.sizeset(mapa[s1]);
76
77        cout << r << endl;
78      }
79    }
80
81  }
```

../dfs.cpp

```
1  //https://www.youtube.com/watch?v=vfK3RtS50vQ
2  void dfs(int u){
3    seen[u] = true;
4    d[u] = tempo++; //tempo de descoberta
5    for(int v: adj[u]){ //iteracao por todos os vertices da lista
6      //de adjacencia
7      if(!seen[v]){ //nao foi visto ainda
8          p[v] = u; //pai de v == u
9          dfs(v); //dfs para ele
10     }
11   }
12   f[u] = tempo++; //tempo de finalizacao
13 }
```

../disktra.cpp

```
1  int dist[100010];
2  vector<pair<int, int>> grafo[100010]; // significa qe cada pair segura o o
       no e a distancia
```

```
3
4  void dijkstra(int s){
5    memset(dist, INF, sizeof(dist));
6    dist[s] = 0;
7    priority_queue<pair<int, int>, // tipo ( pair de distancia e no)
8                   vector<pair<int, int>>, // container
9                   greater<pair<int,int>> > pq; // comparador -> crescente
10                  // na priority_queue a distancia    o proximo
11   pq.push(make_pair(0, s));
12
13   while(!pq.empty()){ // distancia da origem at   o cara que estou ->
        caminho todo
14     pair<int, int> topo = pq.top();
15     pq.pop();
16     int distancia = topo.first;
17     int vertice   = topo.second;
18     if(distancia > dist[vertice]) continue; // se ja tiver uma distancia
          melhor, passa direto!
19
20     for(int i = 0; i < grafo[vertice].size(); i++){
21       int destino = grafo[vertice][i].first;
22       int custo   = grafo[vertice][i].second;
23       if(dist[destino] > dist[vertice] + custo){
24         dist[destino] = dist[vertice] + custo;
25         pq.push(make_pair(dist[destino], destino));
26       }
27     }
28
29   }
30
31 }
32
33
34 int main(){
35   int v, u, w;
36   scanf("%d %d %d", &v,&u,&w);
37   grafo[v].push_back(make_pair(u,w));
38   grafo[u].push_back(make_pair(v,w));
39
40 }
```

../e.cpp

```
1  #define NORTH 0
2  #define EAST 1
3  #define SOUTH 2 //oest
4  #define WEST 3 //lest
5  pair<int, int> mod[5]; // modificador de dire   o na posicao
6  char nomeDir[] = {'N', 'E', 'S', 'W'};
7
8  bool morreu[60][60];
9  pair<int, int> pos; // posicao atual
```

```
10 int n, m, dir;
11
12 bool anda(){ // se n morreu retorna verdade caso contrario false
13   pair<int, int> npos; // nova posicao
14   npos.first = pos.first + mod[dir].first;
15   npos.second = pos.second + mod[dir].second;
16
17   if(npos.first < 0 || npos.first > n || npos.second < 0 || npos.second >
       m){ // caiu
18     if(morreu[pos.first][pos.second]) return true;
19     morreu[pos.first][pos.second] = true;
20     return false;
21   }
22
23   pos = npos;
24   return true;
25 }
26
27
28 int main(){
29   map<char, int> mapa;
30   mapa['N'] = NORTH;
31   mapa['E'] = EAST;
32   mapa['S'] = SOUTH;
33   mapa['W'] = WEST;
34
35   mod[NORTH] = make_pair(0, 1); // soma 1 no y
36   mod[SOUTH] = make_pair(0, -1); //tira 1 no y
37   mod[EAST]  = make_pair(1, 0);
38   mod[WEST]  = make_pair(-1, 0);
39
40   scanf("%d %d", &n,&m);
41   char d;
42   while(scanf("%d %d %c", &pos.first, &pos.second, &d)!=EOF){
43     dir = mapa[d];
44     string s;
45     cin >> s;
46     bool ok = true;
47
48     for(int i = 0; i < s.size(); i++){
49       if(s[i] == 'R'){
50         dir++; // sentido horario
51         dir %= 4;
52       }else if(s[i]=='L'){
53         dir--; // sentido anti-horario
54         if(dir<0) dir += 4;
55       }else{
56         ok = anda();
57         if(!ok) break;
58       }
59     }
60
61     printf("%d %d %c", pos.first, pos.second, nomeDir[dir]);
62     printf("%s", ok? "\n": " LOST \n");
```

```
63
64    }
65  }
```

../ehFibonacci.cpp

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5
6  bool ehFibonacci(int, int);
7
8  int main(){
9    int a, b;
10   cin >> a >> b;
11   string resp = ehFibonacci(a,b) ? "pertencem a fibonacci" :
12                                    "   pertencem a fibonacci";
13   cout << resp << endl;
14 }
15
16 bool ehFibonacci(int a, int b){
17   int t1 = 0, t2 = 1,soma=0;
18   while(soma < 1000){
19     if(t1==a && t2==b)
20       return true;
21     else{
22       soma = t1 + t2;
23       t1 = t2;
24       t2 = soma;
25       cout << soma << endl;
26     }
27   }
28   return false;
29 }
```

../falha.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string n;
6  char d;
7
8  void removeChar(){
9      n.erase(remove(n.begin(),n.end(), d), n.end());
10     //removendo cada caracter que bate da string
11     cout <<atol(n.c_str()) << endl;   //string para long
12 }
```

```
13
14 int main(){
15     while(cin >> d >> n){
16         int zero = (int)d - '0';
17         //char to int '0' de onde a tabela ASCII come a
18        if(!zero && n.compare("0"))
19            break;
20         else
21             removeChar();
22
23     }
24 }
```

../fatorial-com-taxa-erro.cpp

```
1  #include <iostream>
2
3  using namespace std;
4  float e();
5  int main(){
6
7    cout << e() << endl;
8    return 0;
9
10 }
11
12 float e(){
13   int i=2;
14   float termo=3, fatorial=1, eSoma=2;
15   //pq fat de 0    1 e fat de 1    1 logo termo eSoma = 2
16   while (termo>0.0000001){
17     fatorial *= i++;
18     termo = 1/fatorial;
19     eSoma += termo;
20   }
21   return eSoma;
22 }
```

../fatorial-iterativo.cpp

```
1  #include<iostream>
2  #include<stdio.h>
3
4  using namespace std;
5
6  float fatorial(float);
7
8  int main(){
9    float n;
10   cin>>n;
```

```
11    cout << fatorial(n) ;
12    return 0;
13  }
14
15  float fatorial (float n){
16    float soma =1;
17    for(int i=2;i<=n;i++)
18      soma*=i;
19    return 1/soma;
20  }
```

../f-busca-binaria.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   int r[100010];
5   int n;
6
7   bool simula(int k){
8     for(int i=1; i< n; i++){
9       int dif = r[i]- r[i-1];
10      if(dif > k) return false;
11      if(dif == k) k--;
12    }
13    return true;
14  }
15
16  int main(){
17    int t;
18    scanf("%d", &t);
19    for(int caso=1; caso <= t; caso++){
20      int n;
21      scanf("%d", &n);
22      for(int i=0; i < n; i++){
23        scanf("%d", &r[i]);
24      }
25
26      int low = 1, high = (int)(1e8), k; // tenho que inicializar sempre meu
            high e low, minino e maximo
27      //valor muito alto pra high e muito baixo para low
28      while(low <= high){ // busca binaria
29        //testa ambos os casos, caso os dois lados nao respondam, dai o
              valor anterior
30        int mid = (low + high)/2;
31        if(simula(mid)){
32          k = mid;
33          high =  mid - 1;
34        }else {
35          low = mid + 1;
36        }
37      }
```

```
38      printf("Case %d: %d\n",caso, k);
39
40    }
41  }
```

../fibo.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define lli long long int
5
6  lli f[45];
7
8  lli fibo(int n){
9      if(n<=2) return 1;
10     if(f[n]!=-1) return f[n];
11     return f[n] = fibo(n-1)+fibo(n-2);
12 }
13
14
15 int main(){
16
17     memset(f,-1, sizeof f);
18     int a , b;
19     for (int i = 0; i < 45; ++i)    {
20         f[i] = fibo(i);
21     }
22     while(cin >>  a  >> b){
23         lli soma = 0;
24         for (int i = a; i <= b; ++i)
25         {
26             soma += f[i];
27         }
28         cout << soma << endl;
29     }
30
31 }
```

../g-pilha-fila.cpp

```
1  int main(){
2    int n;
3    while(scanf("%d", &n) && n){
4      queue<char> in, out;
5      stack<char> pilha;
6      char c;
7      for(int i=0; i < n; i++){
8        scanf(" %c", &c); //ignora qualquer tipo de espa o em branco!
9        in.push(c);
```

```
10      }
11
12      for(int i=0; i < n; i++){
13        scanf(" %c", &c);
14        out.push(c);
15      }
16
17      while(!in.empty()){
18        pilha.push(in.front());
19        printf("I");
20        in.pop();
21
22        while(!pilha.empty()){
23          if(pilha.top()==out.front()){
24            printf("R");
25            pilha.pop();
26            out.pop();
27          }
28          else break;
29        }
30      }
31      printf("%s", out.empty()? "\n" : " Impossible \n" );
32
33    }
34  }
```

../g-segment-tree.cpp

```
1  //recursivo
2  //sempre em cima de um vetor
3  //pre processada
4  // SEG TREE DE MAXIMO
5  int tree[4*100000+1]; // sempre 4 vezes o numero de elementos q vai ter no
       vetor
6  int a[100010];
7
8  // CONSTRU  O DA SEGTREE
9  void build(int no, int left, int right){
10   // no -> cada segmento, que    uma CAIXINHA, um intervalo
11   // left, right intervalo - limite da esquerda, limite da direita
12   int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha
13   int mid = (left + right)/2;
14   if(left == right){ // chegou na base
15     tree[no] = a[right]; // maior    ele, entao retorna ele mesmo
16     return;
17   }
18   build(filhoEsq, left, mid);
19   build(filhoDir, mid+1, r);
20   tree[no] = max(tree[filhoEsq], tree[filhoDir])
21 }
22
23 // MAIOR ELEMENTO DO INTERVALO
```

```
int query(int no, int left, int right, int i, int j){ // procurando
// no -> no que to. caixinhas
// left, right -> intervalo que ele    responsavel. i, j -> qal intervalo
    que qer o maximo
  if(right < i || left > j) return -INF;
  // se estiver totalmente fora, pra frente, retorne o elemento neuto
  if(left >= i || right <= j) return tree[no];
  // completamente dentro do intervalo que estou procurando
  int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha para
     qual a proxima caixinha ir e abrir
  int mid = (left + right)/2;
  int maxEsq = query(filhoEsq, left, mid, i, j);
  int maxDir = query(filhoDir, mid + 1, right, i, j);
  return max(maxEsq, maxDir);
}

// ALTERA VALOR NA ARVORE DE SEGUIMENTO E REPASSA A ALTERA  O
void update(int no, int left, int right, int pos, int val){
  if(pos > right || pos < left) return ; // fora do meu intervalo
  if(left == right){
    // chegou no lugar que qer fazer o update
    tree[no] = val;
    return;
  }
  int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha
  int mid = (left + right)/2;
  update(filhoEsq, left, mid, pos, val);
  update(filhoDir, mid + 1, right, pos, val);
  tree[no] = max(tree[filhoEsq], tree[filhoDir]);
}

int main(){
  int N, k;
  while(scanf("%d %d", &N, &K) != EOF && N){
    for(int i = 1; i <= N; i++){
      scanf("%d", &a[i]);
    }
    build(1, 1, N); // sempre constroi do primeiro seguimento, porque ele
       vai conter todo o intervalo
    for(int i = 0; i < K; i++){
      char c;
      int a, b;
      scanf(" %c %d %d", &c,&a,&b);
      if(c == 'C'){ // change
        // no inicial, limite esquerda, limite direira
        // posicao, valor
        update(1, 1, N , a, b); // update no ponto a e qer colocar o valor
            b
      }else{
        // no inicial, limite esquerda, limite direita
        // esquerda do intervalo, direita do intervalo
        int v = query(1, 1, N, a, b);
        c = '0';
        if(v > 0) c = '+';
```

```
74          else if(v < 0) c = '-';
75          printf("%c\n", c);
76        }
77      }
78
79    }
80    return 0;
81  }
```

../h.cpp

```
1   int main(){
2     int n;
3     scanf("%d", &n);
4     while(n--){
5       pair<long long,long long> left, right, mid;
6       left = make_pair(0, 1);
7       right = make_pair(1, 0);
8       mid = make_pair(1,1);
9       string str;
10      cin >> str;
11      for(int i=0; i < str.size(); i++){
12        if(str[i] == 'L'){
13          right = mid;
14          mid.first += left.first;
15          mid.second += left.second;
16        }else{
17          left = mid;
18          mid.first += right.first;
19          mid.second += right.second;
20        }
21      }
22      printf("%lld/%lld\n",mid.first, mid.second);
23    }
24  }
```

../i.cpp

```
1
2   int n, m;
3   char grid[110][110];
4   bool vis[110][110];
5
6   void dfs(int x, int y){
7     if(x < 0 || x >= n
8       || y < 0 || y >=m
9       || grid[x][y] !='@'
10      || vis[x][y]) return;
11
12    vis[x][y] = true;
```

```
13
14    for(int i =-1; i <= 1; i++){
15      for(int j = -1; j <=; j++){
16        dfs(x+i, y+j);
17      }
18    }
19
20  }
21
22
23  int main(){
24    while(scanf("%d %d", &n, &m) && (n|m)){
25      int cont = 0;
26      memset(vis, 0, sizeof(vis));
27      for(int i = 0; i < n; i++)
28        scanf("%s", grid[i]);
29      for(int i = 0; i < n; i++){
30        for(int j = 0; j < m; j++){
31          if(grid[i][j]!='@') continue;
32          if(vis[i][j]) continue;
33          dfs(i,j);
34          cont++;
35        }
36      }
37
38    }
39  }
```

../mdc.cpp

```
1  int mdc(int a, int b) {
2      return b == 0 ? a : mdc(b, a % b);
3  }
```

../passelivre.cpp

```
1  #include <bits/stdc++.h> //todo as biblios
2
3  struct pessoa{
4      int lugar;
5      std::string nome;
6  };
7
8  int pessoas;
9  pessoa people[100010];
10
11 void ordena(pessoa p, int pp ){
12     if(pp==pessoas-1) return;
13
14     if(p.lugar>people[pp+1].lugar){
```

```
15          pessoa temp = people[pp+1];
16          people[pp+1] = p;
17          people[pp] = temp;
18      }
19      ordena(people[pp+1],pp+1);
20  };
21
22  int main(){
23      int filas;
24      scanf("%d", &filas);
25
26      while(filas--){
27
28          scanf("%d", &pessoas);
29          for(int i=0; i<pessoas;i++){
30              std::cin >> people[i].nome >> people[i].lugar;
31          }
32
33          ordena(people[0], 0);
34
35          for(int i=0; i<pessoas-1;i++){
36            std::cout << people[i].nome << ' ';
37          }
38          std::cout << people[pessoas-1].nome << std::endl;
39      }
40  }
```

../promocao.cpp

```
1   #include <bits/stdc++.h>
2   // PD -> dicas : numero muito grande  N  O  FUNCIONA
3   // numero muito pequena PD cabulosa
4   using namespace std;
5
6   int preco[110], peso[110];
7   int m[110][1010]; //PD esta na memorizacao (DISP por ir de 0--1000)
8
9   int t; //ultimo item da lista
10
11  int best(int item, int disp){ //index do item e capacidade disponivel
12      //CASO BASE SEMPRE PRIMEIRO
13      if(item == t) return 0; //ja passou -> BORA PARAR
14      if(m[item][disp] != -1) return m[item][disp]; //ESTADOS
15
16      int ans = best(item+1, disp); //nao quero o item atual
17      if(disp >= peso[item]){ //se eu quiser o iten preciso ter capacidade
18          ans = max(ans, best(item+1, disp - peso[item])+preco[item]);
19          //verifica quem vai me dar mais lucro , diminuindo minha
                capacidade
20          //pega o item atual e tenta pegar mais (os proximos)
21      }
22      return m[item][disp] = ans; //memorizando BUIA
```

```
}

int main(){
    int capacidadeMochila;

    while((scanf("%d", &t)), t){
        memset(m,-1, sizeof(m)); //setando geral como -1 NAO ROLA COM
            VECTOR

        for (int i = 0; i < t; ++i) {
            scanf("%d %d", &preco[i], &peso[i]);
        }
        scanf("%d", &capacidadeMochila);

        printf("%d\n", best(0, capacidadeMochila)); //primeiro item e a
            capacidade da mochila
    }
    return 0;
}
```

../resolver.cpp

```
#include <cstdio>

int main(){
    long long int n1,n2,n3;
    while((scanf("%lld %lld %lld", &n1,&n2,&n3)==3)){
        if(n1==0 && n2==0 && n3==0)
            break;
        if(n3%10 != 6 && n3%10 !=1)
            printf("N o Achou.\n");
        else
            printf("Achou.\n");
    }
}
```

../segment-tree-maximos.cpp

```
//recursivo
//sempre em cima de um vetor
//pre processada
// SEG TREE DE MAXIMO
int tree[4*100000+1]; // sempre 4 vezes o numero de elementos q vai ter no
    vetor
int a[100010];

// CONSTRU  O DA SEGTREE
void build(int no, int left, int right){
  // no -> cada segmento, que    uma CAIXINHA, um intervalo
  // left, right intervalo - limite da esquerda, limite da direita
```

```
12    int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha
13    int mid = (left + right)/2;
14    if(left == right){ // chegou na base
15      tree[no] = a[right]; // maior    ele, entao retorna ele mesmo
16      return;
17    }
18    build(filhoEsq, left, mid);
19    build(filhoDir, mid+1, r);
20    tree[no] = max(tree[filhoEsq], tree[filhoDir])
21  }
22
23  // MAIOR ELEMENTO DO INTERVALO
24  int query(int no, int left, int right, int i, int j){ // procurando
25  // no -> no que to. caixinhas
26  // left, right -> intervalo que ele    responsavel. i, j -> qal intervalo
        que qer o maximo
27    if(right < i || left > j) return -INF;
28    // se estiver totalmente fora, pra frente, retorne o elemento neuto
29    if(left >= i || right <= j) return tree[no];
30    // completamente dentro do intervalo que estou procurando
31    int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha para
          qual a proxima caixinha ir e abrir
32    int mid = (left + right)/2;
33    int maxEsq = query(filhoEsq, left, mid, i, j);
34    int maxDir = query(filhoDir, mid + 1, right, i, j);
35    return max(maxEsq, maxDir);
36  }
37
38  // ALTERA VALOR NA ARVORE DE SEGUIMENTO E REPASSA A ALTERA  O
39  void update(int no, int left, int right, int pos, int val){
40    if(pos > right || pos < left) return ; // fora do meu intervalo
41    if(left == right){
42      // chegou no lugar que qer fazer o update
43      tree[no] = val;
44      return;
45    }
46    int filhoEsq = no * 2, filhoDir = no * 2 + 1; // fazendo a sominha
47    int mid = (left + right)/2;
48    update(filhoEsq, left, mid, pos, val);
49    update(filhoDir, mid + 1, right, pos, val);
50    tree[no] = max(tree[filhoEsq], tree[filhoDir]);
51  }
52
53  int main(){
54    build(1, 1, N);
55    return 0;
56  }
```

../spoj-energia-bfs.cpp

```
1  //bfs (busca em largura - lado pra dps descer FILA) e dfs(busca em
       profundidade - fundo pra dps ladear PILHA-recurs o)
```

```
2
3  #define push_back pb
4  #define push ps
5  #define pop pp
6
7
8  #include <bits/stdc++.h>
9  using namespace std;
10
11 vector<int> grafo[110]; // vetor de vector -> lista de adjacencia
12 int vis[110]; // visita  es
13
14
15 int bfs(int s){
16   queue<int> q;
17   vis[s] = 1; // marca como visitado
18   int cont = 1;
19   q.ps(s); // origem
20   while(!q.empty()){
21     int a = q.front();
22     q.pp();
23     for(int i=0; i< grafo[a].size(); i++){
24       int b = grafo[a][i]; // percorrendo o vector de adjacents
25       if(vis[b]) continue; // se ja visitou, passa direto
26       cont++;
27       vis[b] = 1;
28       q.ps(b);
29     }
30   }
31
32   return cont;
33 }
34
35 int main(){
36   int v, e, caso = 1;
37   while(scanf("%d %d", &v, &e) && (v|e)){
38     memset(vis, 0, sizeof(vis));
39     for(int i = 0; i <= v; i++){
40       grafo[i].clear();
41     }
42     for(int i = 0; i < e; i++){
43       int a,b;
44       scanf("%d %d", &a, &b);
45       grafo[a].pb(b);
46       grafo[b].pb(a);
47     }
48     printf("Teste %d\n", caso++);
49     printf("%s\n\n", bfs(1) == v? "normal" : "falha");
50   }
51
52   return 0;
53 }
```

../spoj-energia-dfs.cpp

```cpp
//bfs (busca em largura - lado pra dps descer FILA) e dfs(busca em
    profundidade - fundo pra dps ladear PILHA-recurs o)

#define push_back pb
#define push ps
#define pop pp


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

vector<int> grafo[110]; // vetor de vector -> lista de adjacencia
int vis[110]; // visita  es
int cont;

void dfs(int no){ // vertice que estou indo visitar
  vis[no] = 1;
  cont++;
  for(int i = 0; i < grafo[no].size(); i++){
    int b = grafo[no][i];
    if(!vis[b]) dfs(b);
  }
}

int main(){
  int v, e, caso = 1;
  while(scanf("%d %d", &v, &e) && (v|e)){
    memset(vis, 0, sizeof(vis));
    cont = 0;
    for(int i = 0; i <= v; i++){
      grafo[i].clear();
    }
    for(int i = 0; i < e; i++){
      int a,b;
      scanf("%d %d", &a, &b);
      grafo[a].pb(b);
      grafo[b].pb(a);
    }
    dfs(1);
    printf("Teste %d\n", caso++);
    printf("%s\n\n", cont == v ? "normal" : "falha");
  }

  return 0;
}
```

../streams.cpp

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

```
3
4  int hash(int x){
5      return ((3*x)+1)%7;
6  }
7
8  int main(){
9      string linha;
10     int x;
11     char virgula;
12     vector<int> nums; //vetor de inteiro
13     stringstream xx;
14     bitset<8> bitmap;
15     bitset<8> num; //vai receber o hashed
16
17     while(cin >> linha){
18         xx << linha;
19         while(xx >> x){ //come o proximo int
20             nums.push_back(x);
21             xx >> virgula; //come o proximo char (virgulas)
22         }
23         for (int i = 0; i < nums.size(); ++i){
24             int j = hash(nums[i]);
25             if(j){
26                 num = j;
27                 for (int z = num.size()-1; z > 0; ++z){
28                     if(num[z]){
29                         bitmap[z] = 1;
30                         break;
31                     }
32                 }
33             }
34         }
35         cout << bitmap << endl;
36
37         nums.clear(); //limpa vetor de inteiro
38         xx.clear(); //limpa a stream e suas flags
39         bitmap.reset(); //reseta bitmap
40         num.reset();
41     }
42 }
```

../unionfind.cpp

```
1  class UnionFind{
2    public:
3      vector<int> p, rank; // rank == altura do conjunto i
4      UnionFind(int n){ // iniciais
5        p.resize(n); // quem    pai de qem?
6        rank.assign(n,0); // inicializa o vector com tamanho n e valor 0
7        for(int i=0;i<n;i++){
8          p[i] = i; // inicialmente eu sou meu pai
9        }
```

```
10      }
11
12      int findSet(int i){ // quem manda mesmo nele
13        if(p[i] == i) return i;
14        return (p[i] = findSet(p[i])); // quem manda no meu pai agora manda
              em mim
15      }
16
17      bool isSameSet(int i,int j){ // fazem parte do mesmo conjunto
18        return findSet(i) == findSet(j);
19      }
20
21      void unionSet(int i, int j){ // juntando caras
22        if(isSameSet(i,j)) return; // se eles fazem parte do mesmo conjunto,
              ja retorno
23
24        int x = findSet(i); // acho pai de i
25        int y = findSet(j); // acho pai de j
26
27        if(rank[x] > rank[y]){ // maior arvore (quem manda em mais gente?)
28          p[y] = x;
29          return;
30        }
31
32        p[x] = y; // escolha arbitr ria (tamanho da arvore de y pode ser
              maior ou igual ao tamanho da arvore de x)
33        if(rank[x] == rank[y]) rank[y]++; // se as arvores forem do mesmo
              tamanho, alguem tem q ficar em cima, logo tamanho aumentou
34      }
35
36  };
```

../uri-1022.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   int main(){
5     int cases;
6     scanf("%d", &cases);
7     for(int i=0; i<cases;i++){
8       int n1, d1,n2,d2;
9       int v1,v2;
10      char op;
11      scanf("%d / %d %c %d / %d",&n1,&d1,&op,&n2,&d2);
12
13      switch(op){
14        case '+':
15          v1 = n1*d2+n2*d1;
16          v2 = d1*d2;
17        break;
18        case '-':
```

```
19          v1 = n1*d2-n2*d1;
20          v2 = d1*d2;
21        break;
22        case '/':
23          v1 = n1*d2;
24          v2 = n2*d1;
25        break;
26        case '*':
27          v1 = n1*n2;
28          v2 = d1*d2;
29        break;
30      }
31
32      printf("%d/%d = ",v1,v2);
33
34      /*
35      Dados dois ou mais n meros , se um deles    divisor de todos os outros
           , ent o ele    o MDC dos n meros dados
36      */
37
38      int num=-1;
39      int maior=-1;
40      if(v1>v2)
41        maior=v1;
42      else maior = v2;
43
44      for(int i=1;i<=maior;i++){
45        if(v1%i==0 && v2%i==0)
46            num = i;
47      }
48
49      if(num!=-1){
50        v1 /= num;
51        v2 /= num;
52      }
53      printf("%d/%d\n",v1,v2);
54    }
55
56  }
```

../uri-1024.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   void primeira(string &texto){
5     for(int i=0;i<texto.size();i++){
6       if(isalpha(texto[i])){ //is alphabetic(letra)
7         texto[i] += 3;
8       }
9     }
10  }
```

```
11
12 void segunda(string &texto){
13   reverse(texto.begin(), texto.end());
14 }
15
16 void terceira(string &texto){
17   for(int i=texto.size()/2;i<texto.size();i++){
18     texto[i]--;
19   }
20 }
21
22 int main(){
23   int t;
24   scanf("%d", &t);
25   cin.ignore();
26   while(t--){
27     string texto;
28     getline(cin,texto);
29     primeira(texto);
30     segunda(texto);
31     terceira(texto);
32     cout << texto << endl;
33   }
34   return 0;
35 }
```

../uri-1029.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int calls;
5
6  int fibo(int n){
7    calls++;
8    if(n<=1) return n;
9    return fibo(n-1) + fibo(n-2);
10 }
11
12 int main(){
13   int cases, n;
14   scanf("%d", &cases);
15
16   for(int i=0; i<cases; i++){
17     calls=-1; //tirando a primeira chamada
18     scanf("%d", &n);
19     printf("fib (%d) = %d calls = %d\n", n, calls, fibo(n) );
20   }
21 }
```

../uri-1047.cpp

```cpp
#include <cstdio>

using namespace std;

int main(){
  int hin, hfin, min, mfin;

  while( scanf("%d %d %d %d", &hin, &min, &hfin, &mfin) == 4){
  // scanf("%d %d %d %d", &hin, &min, &hfin, &mfin);

    int comeco = hin * 60 + min; //transforma para minuto
    int final  = hfin * 60 + mfin;

    if(hin <= hfin){
      int duracao = final - comeco;

      if(duracao == 0)
       printf("O JOGO DUROU %d HORA(S) E %d MINUTO(S)\n",
          24, duracao%60);
      else
        printf("O JOGO DUROU %d HORA(S) E %d MINUTO(S)\n",
          duracao/60, duracao%60);
    }else{
        int duracao = (24*60 - comeco) + final;
        printf("O JOGO DUROU %d HORA(S) E %d MINUTO(S)\n",
        duracao/60, duracao%60);
    }


  }




}
```

../uri-1051.cpp

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

int main(){
    float sal;
    while(cin>> sal){
        // scanf("%f", &sal);

        if(sal <=2000)
            printf("Isento\n");
        else{
```

```
12              float final;
13              if(sal>=2000.01 && sal <= 3000){
14                  final = (sal - 2000) * 0.08;
15              }else if(sal >= 3000.01 && sal <= 4500){
16                  final = (sal - 3000)* 0.18 + (1000 * 0.08);
17              }else {
18                  final = ((sal - 4500) * 0.28) + (1500 * 0.18) + (1000 *
                        0.08);
19              }
20              printf("R$ %.2f\n", final);
21          }
22      }
23
24
25  }
```

../uri-1052.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   int main(){
5       int mes;
6       scanf("%d", &mes);
7
8       switch(mes) {
9           case 1:
10              printf("January\n");
11          break;
12          case 2:
13              printf("February\n");
14          break;
15          case 3:
16              printf("March\n");
17          break;
18          case 4:
19              printf("April\n");
20          break;
21          case 5:
22              printf("May\n");
23          break;
24          case 6:
25              printf("June\n");
26          break;
27          case 7:
28              printf("July\n");
29          break;
30          case 8:
31              printf("August\n");
32          break;
33          case 9:
34              printf("September\n");
```

```
35          break;
36          case 10:
37              printf("October\n");
38          break;
39          case 11:
40              printf("November\n");
41          break;
42          case 12:
43              printf("December\n");
44          break;
45      }
46  }
```

../uri-1061.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5
6  }
```

../uri-1064.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      float num;
6      int pos = 0;
7      float media = 0;
8    for(int i=0; i <6; i++){
9      scanf("%f", &num);
10     if(num > 0){
11         pos++;
12         media+=num;
13     }
14   }
15
16   printf("%d valores positivos\n%.1f\n",pos, media/pos );
17 }
```

../uri-1065.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
```

```
    int num;
    int pos = 0;
  for(int i=0; i <5; i++){
    scanf("%d", &num);
    if(!(num % 2))
        pos++;

  }

  printf("%d valores pares\n",pos);
}
```

../uri-1068.cpp

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

int main(){
  string ex;
  while(cin >> ex){
    int par = 0;
    for(int i=0; i < ex.size(); i++){
      if(ex[0] == ')'){
        par = -1;
        break;
      }

      if(ex[i] == '('
        par++;
      else if(ex[i] == ')'
        par--;

      if(par < 0)
        break;
    }

    if(par == 0)
      printf("correct\n");
    else printf("incorrect\n");
  }
}
```

../uri-1069.cpp

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

int main(){
  string l;
  int casos;
```

```
  scanf("%d", &casos);
  while(casos--){
    cin >> l;
    int di = 0;
    int abriu = 0, fechou = 0;

    for(int i=0; i < l.size(); i++){
      if(l[i]=='.'){
        continue;
      }else if(l[i]=='<'){
        abriu++;
      }else if(l[i]=='>'){
        if(abriu > 0){
          fechou++;
          di++;
          abriu--;
        }
      }

    }

    printf("%d\n", di);

  }
}
```

../uri-1070.cpp

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

int main(){
    int num;
    int pos = 5;
        scanf("%d", &num);
    do{
        int g = num++;
        if(g%2){
            printf("%d\n", g);
            pos--;
        }
    }while(pos>-1);
}
```

../uri-1110.cpp

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

queue<int> cartas;
```

```
5
6  int ff(){
7    int k = cartas.front();
8    cartas.pop();
9    return k;
10 }
11
12 int main(){
13   int c;
14
15   while(scanf("%d", &c)==1){
16     if(c==0)
17       break;
18
19     for(int i=1; i<=c; i++){
20       cartas.push(i);
21     }
22
23      printf("Discarded cards: ");
24
25     int f;
26     while(cartas.size() > 2){
27       f = cartas.front();
28       printf("%d, ", f);
29       cartas.pop();
30       cartas.push(ff());
31     }
32
33     printf("%d\n", ff());
34
35     printf("Remaining card: %d\n", ff());
36     queue<int>().swap(cartas);
37   }
38
39   return 0;
40 }
```

../uri-1153.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int fatorial(int n){
5    if(n <= 1)
6      return 1;
7    else return n * fatorial(n-1);
8  }
9
10 int main(){
11   int c;
12   scanf("%d", &c);
13   printf("%d\n", fatorial(c));
```

```
}
```

../uri-1259.cpp

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

int main(){
  int c, n;
  scanf("%d", &c);
  vector<int> par;
  vector<int> im;

  while(c--){
    scanf("%d", &n);
    n = n < 0 ? -n : n;

    if(n%2==0)
      par.push_back(n);
    else
      im.push_back(n);
  }

  sort(par.begin(), par.end());
  sort(im.begin(), im.end(), greater<int>());

  // for (std::set<int>::iterator it=im.begin(); it!=im.end(); ++it)
  //   std::cout << *it << endl;

  // for (std::set<int>::iterator it=par.begin(); it!=par.end(); ++it)
  //   std::cout << *it << endl;

  for(auto i: par){
    printf("%d\n", i);
  }
  for(auto i: im){
    printf("%d\n", i);
  }

  return 0;
}
```

../uri-1281.cpp

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

int main(){
  int cas;
  scanf("%d", &cas);
```

```
7     map<string,float> prods; //nome do produto (KEY) valor (VALUE)
8
9     while(cas--){
10      int tamLista;
11      string nomeProd;
12      float valProd;
13
14      scanf("%d", &tamLista);
15
16      for(int i=0;i<tamLista;i++){
17        cin >> nomeProd >> valProd;
18        prods[nomeProd] = valProd; //definindo a chave e o valor pra ela
19      }
20
21      scanf("%d", &tamLista);
22
23      float soma = 0;
24
25      int qtd;
26      for(int i=0;i<tamLista;i++){
27        cin >> nomeProd >> qtd;
28        valProd = prods.find(nomeProd) -> second; //chave (FIRST) valor (
              SECOND)
29        soma += (qtd*valProd);
30      }
31
32      printf("R$ %.2f\n", soma );
33      prods.clear();
34
35    }
36  }
```

../uri-2134.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   int main(){
5     int povo, t=1;
6     while(cin >> povo){
7       printf("Instancia %d\n", t);
8       string nome;
9       int prob;
10      set<pair<int,string>> alunos; //ordena por chave e dps por valor
11      for(int i=0;i<povo;i++){
12        cin >> nome >> prob;
13        alunos.insert(make_pair(prob,nome));
14      }
15
16      int n = alunos.begin() -> first; //numero
17      string reprovado;
18
```

```
19      bool flag = true;
20      for(auto &c : alunos){
21        if(c.first==n){
22          flag = false;
23          reprovado = c.second;
24        }else
25          break;
26       }
27
28      if(!flag)
29        cout << reprovado;
30      else
31        cout << alunos.begin() -> second; //nome
32
33      printf("\n\n");
34
35      t++;
36    }
37
38  }
```

../uri-2235.cpp

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   int main(){
5     int a,b,c, presente = 2016;
6     while(scanf("%d %d %d", &a,&b,&c)==3){
7       if((presente+a-b)==presente)
8         printf("S\n");
9       else if((presente+a-c)==presente)
10        printf("S\n");
11      else if((presente+b-a)==presente)
12        printf("S\n");
13      else if((presente+b-c)==presente)
14        printf("S\n");
15      else if((presente+c-a)==presente)
16        printf("S\n");
17      else if((presente+c-b)==presente)
18        printf("S\n");
19      else if((presente+a+b-c)==presente)
20        printf("S\n");
21      else if((presente+a+c-b)==presente)
22        printf("S\n");
23      else if((presente+b+c-a)==presente)
24        printf("S\n");
25      else if((presente+b+a-c)==presente)
26        printf("S\n");
27      else if((presente+c+a-b)==presente)
28        printf("S\n");
29      else if((presente+c+b-a)==presente)
```

```
30        printf("S\n");
31      else
32        printf("N\n");
33    }
34
35  }
```

../uva-10050.cpp

```
1   #include <iostream>
2   #include <bitset>
3   #include <vector>
4   using namespace std;
5
6   int harsals(int, vector<int>, int);
7
8   int main(){
9       vector<int> H;
10      int T, N, P, h;
11      while(cin >> T){ //T casos de teste | N numero de dias | P qtd
            partidos | h passo
12          while(T--){
13              cin >> N >> P;
14              for(int i=1; i<=P;i++){
15                  cin >> h;
16                  H.push_back(h);
17              }
18              cout << harsals(N,H,P)<< endl;
19              H.clear();
20          }
21      }
22      return 0;
23  }
24
25  int harsals(int N, vector<int> H, int p){
26      bitset<3651> calendar;
27      calendar.reset();
28      for(int i=1 ; i<=p; i++){
29          int h = H[i-1];
30          for (int j=h; j<=N; j+=h)   {
31              calendar.set(j, true); //muda o bit da posi   o atual
32          }
33      }
34      for(int i=6;i<=N;i+=7)
35          calendar.reset(i);
36      for(int i=7;i<=N;i+=7)
37          calendar.reset(i);
38
39      return calendar.count();
40
41  }
```

../uva-100.cpp

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

int threeNPlusOne(int i, int j);

int main(){
    int i, j;
    while(cin >> i >> j) { //"entrando no i"
        int max = threeNPlusOne(i,j);
        cout << i <<" "<< j << " "<< max << endl;
    }
    return 0;
}

int threeNPlusOne(int i, int j){
    if (i > j ){
        int temp = i;
        i = j;
        j = temp;
    }
    int max = 1;
    for( int go = i; go <= j ; go++ ){
        int ciclo = 1;
        int x = go;
        while( x>1 ){
            if( ( x%2 ) == 0 )  x = x/2;
            else    x = 3*x + 1;
            ciclo++;
        }

        if( max<ciclo) max = ciclo;
    }
    return max;

}
```

../uva-10189.cpp

```cpp
#include<iostream>
#include<vector>
using namespace std;

void minesweeper( int , int , vector<vector<int> > &);
void incrementa( int & );

int main(){
    int n,m;
    vector<vector<int> > mat;
    while(cin >> n >> m , n!=0 && m!=0 ){
        mat.resize(n+2); //linha
```

```
13          for(int l=1; l<=n; l++){
14              mat[l].resize(m+2, 0); //coluna
15              for(int c=1 ; c<=m ; c++){
16                  char sig;
17                  cin >> sig;
18                  if( sig=='*' )
19                      mat[l][c] = -1;
20                  else mat[l][c] = 0;
21              }
22          }
23          minesweeper(n,m,mat);
24          for(int l=1;l<=n;l++){
25              for (int c=1;c<=m;c++){
26                  int value = mat[l][c];
27                  cout << value;
28              }
29              cout << endl ;
30          }
31
32      }
33
34      return 0;
35  }
36
37  void minesweeper( int n, int o, vector<vector<int> > &m ){
38      for(int i=1; i<=n; i++){
39          for(int j=1 ; j<=o ; j++){
40              if( m[i][j] == -1 ){
41                  incrementa ( m[i][j+1] );
42                  incrementa ( m[i][j-1] );
43                  incrementa ( m[i+1][j+1] );
44                  incrementa ( m[i-1][j-1] );
45                  incrementa ( m[i+1][j] );
46                  incrementa ( m[i-1][j] );
47                  incrementa ( m[i+1][j-1] );
48                  incrementa ( m[i-1][j+1] )  ;
49              }
50          }
51      }
52
53  }
54
55  void incrementa( int &value ){ //recebendo referencia para um inteiro
56      if( value != -1 ) value++;
57  }
```