Makefile

```
CPP = $(addsuffix .cpp, ${dir})
all:
	mkdir ${dir} && touch ${dir}/$(CPP) && touch ${dir}/input && cp Makefile2 $
{dir}/Makefile
	echo "#include<cstdio>\n\nint main(){\n\n    return 0;\n}"  >> ${dir}/$(CPP)
	cd ${dir} && vim ${CPP}
```

Makefile2

```
all:
	g++ ${file} -o $(basename ${file}) && ./$(basename ${file}) < input
```

TARZAN (DFS)

```
#include<cstdio>
#include<math.h>

const int maxn = 1010;
int mtx[maxn][maxn];
int dist_x[maxn];
int dist_y[maxn];
int marcado[maxn];
int n, d;

int dfs(int v){
   marcado[v] = 1;
   for(int i = 1; i<=n; ++i) if (mtx[v][i] && !marcado[i]) dfs(i);
}

int main(){
   int x, y;
   int dist;

   scanf(" %d %d", &n, &d);

   d *= d;

   for (int i=1; i<=n; ++i){
     scanf(" %d %d", &x, &y);
     dist_x[i] = x;
     dist_y[i] = y;
   }

   for (int i = 1; i<=n; ++i){
      for (int j = 1; j<=n; ++j) mtx[i][j] = mtx[j][i] = 0;
      marcado[i] =0 ;
   }

   for(int i = 1; i<=n; ++i){
      for (int j = 1; j<=n; ++j){
         dist = pow(dist_x[i] - dist_x[j], 2) + pow(dist_y[i] - dist_y[j], 2);
         //printf("%d\n\n", dist);
         if (dist <= d) mtx[i][j] = mtx[j][i] = dist;
      }
   }

   dfs(1);

   for(int i = 1; i<=n; ++i)
      if (!marcado[i]){ printf("N\n"); return 0;}

   printf("S\n");
   return 0;
}
```

FRETE08 (AGM)

```
#include<cstdio>

const int maxn = 1010;
const int inf = 0x3f3f3f3f; int grafo[maxn][maxn]; int distancia[maxn]; int marcado[maxn]; int n, m;

int arvore_minima(){
  int custo = 0;
  distancia[0] = 0;

  for (int i = 0; i<n; ++i){
     int minimo = -1;
     int melhor_distancia = inf;

     for (int j=0; j<n; ++j){
        if(!marcado[j] && distancia[j] < melhor_distancia){
           minimo = j;
           melhor_distancia = distancia[j];
        }
     }

     marcado[minimo] = 1;
     custo += distancia[minimo];

     for (int j=0; j<n; ++j){
        if (!marcado[j] && distancia[j] > grafo[minimo][j]){
           distancia[j] = grafo[minimo][j];
        }
     }
  }

  return custo;
}

int main(){
   int a, b, c;
   scanf(" %d %d", &n, &m);

   for(int i=0; i<n; ++i){
      for (int j=0; j<n; ++j){
         grafo[j][i] = inf;
      }
      distancia[i] = inf;
      marcado[i] = 0;
   }
   for(int i=0; i<m; ++i) {
      scanf(" %d %d %d", &a, &b, &c);
      grafo[a][b] = c;
      grafo[b][a] = c;
   }

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

   return 0;
}
```

CORRD11 (qsort)

```
#include<cstdio>
#include<cstdlib>

int carros[110];
int carros_pos[110];

int cmp(const void *n1, const void *n2){
   return (*(int *)n1 - *(int *)n2);
}

int main(){

   int n, m, aux, soma;
   scanf(" %d %d", &n, &m);

   for(int i = 1; i <= n; i++){
      soma = 0;
      for(int j = 1; j <= m; j++){
         scanf(" %d", &aux);
         soma += aux;
      }
      carros[i] = soma;
   }

   for(int i = 1; i <= n; i++){
      carros_pos[i] = carros[i];
   }

   qsort(carros, n+1, 4, cmp);

   int pos1;
   for(int i = 1; i <= n; i++){
      if (carros[1] == carros_pos[i]) pos1 = i;
   }

   printf("%d\n", pos1);

   return 0;
}
```

CHAMADA1 (Ordenar em ordem alfabética)

```
#include<cstdio>
#include<string.h>


int main(){
    int n, k;
    char names[110][25];
    char temp[25];

    scanf(" %d %d", &n, &k);
    for (int i = 1; i<=n; ++i) scanf(" %s", names[i]);

    //for (int i = 1; i<=n; ++i) printf(" %s\n", names[i]);
    //printf("\n");

    for (int i = 1; i<=n; ++i)
        for (int j = 1; j<=n; ++j)
            if (strcmp(names[i], names[j]) < 0){
                strcpy(temp, names[i]);
                strcpy(names[i], names[j]);
                strcpy(names[j], temp);
            }

    //for (int i = 1; i<=n; ++i) printf(" %s\n", names[i]);
    //printf("\n");

    printf("%s\n", names[k]);

    return 0;
}
```

ENERGIA (DFS)

```
#include<cstdio>

const int maxe = 110;
int grafo[maxe][maxe];
int visitados[maxe];
int n_visitados,e;

int dfs(int v){
   visitados[v] = 1;
   ++n_visitados;

   for (int i = 1; i<=e; ++i)
      if (grafo[v][i] && !visitados[i])
         dfs(i);
}

int main(){
   int l, x, y, count=0;

   while (true){
      scanf(" %d %d", &e, &l);
      if (e==0) break;

      printf("Teste %d\n", ++count);

      n_visitados =0;
      for (int i = 1; i<=e; ++i){
         for (int j = 1; j<=e; ++j) grafo[i][j] = grafo[j][i] = 0;
         visitados[i] = 0;
      }

      for (int i = 1; i<=l; ++i)
         scanf(" %d %d", &x, &y), grafo[x][y] = grafo[y][x] = 1;

      dfs(1);

      //printf("%d\n\n", n_visitados);

      if (n_visitados == e) printf("normal\n");
      else printf("falha\n");
      printf("\n");
   }

   return 0;
}
```

FLIPERAM (qsort)

```
#include<cstdio>
#include<cstdlib>

const int maxn = 10100;
int array[maxn];

int cmp(const void *n1, const void *n2){
        return (*(int *)n2 - *(int *)n1);
}

int main(){
    int n, m, n_atual;

    scanf (" %d %d", &n, &m);

    for (int i = 0; i<n; ++i){
        scanf(" %d", &n_atual);
        array[i] = n_atual;
    }

    qsort(array, n, 4, cmp);

    for (int i =0 ; i<m; ++i) printf("%d\n", array[i]);


    return 0;
}
```

FUSOES (Union Find)

```
#include<cstdio>

const int maxn = 100100;
int pai[maxn];
int rank[maxn];
int bancos[maxn];
int n, k;

int find(int a){
   if(pai[a] ==a) return a;
   pai[a] = find(pai[a]);
   return pai[a];
}

int make_union(int a, int b){
   int pai_a = find(a);
   int pai_b = find(b);
   if(rank[pai_a] > rank[pai_b]) pai[pai_b] = pai_a;
   else if(rank[pai_b] > rank[pai_a]) pai[pai_a] = pai_b;
   else pai[pai_b] = pai_a, rank[pai_a]++;
}

int main(){
   char f;
   int a, b;

   scanf(" %d %d", &n, &k);

   for(int i = 1; i<=n; ++i){
      pai[i] = i;
      rank[i] = 0;
   }
   for(int i = 1; i<=k; ++i){
      scanf(" %c", &f);
      scanf(" %d %d", &a, &b);

      if (f =='C'){
         if(find(a) == find(b)) printf("S\n");
         else printf("N\n");
      }
      else{
         make_union(a, b);
      }
   }
   return 0;
}
```

REUNIAO2 (floyd-warshall)

```
#include<cstdio>

const int inf = 0x3f3f3f3f;
const int maxn = 110;
int grafo[maxn][maxn];

int main(){
   int n, m, x, y, z;
   int maior_distancia=-1;
   int menor = inf;

   scanf(" %d %d", &n, &m);

   for(int i=0; i<n; ++i){
      for(int j = 0; j<n; ++j){
         if(i==j) grafo[i][j] = 0;
         else grafo[i][j] = inf;
      }
   }

   for(int i=0; i<m; ++i){
      scanf(" %d %d %d",  &x, &y, &z);
      if(z < grafo[x][y]){
         grafo[x][y] = grafo[y][x] = z;
      }
   }

   for (int k = 0; k<n; ++k){
      for (int i = 0; i<n; ++i){
         for (int j = 0; j<n; ++j){
            if(grafo[i][k] + grafo[k][j] < grafo[i][j]) grafo[i][j] = grafo[i][k] + grafo[k][j];
         }
      }
   }
   for (int i = 0; i<n; ++i){
      for (int j = 0; j<n; ++j)
         if (grafo[i][j] > maior_distancia) maior_distancia = grafo[i][j];

      if(maior_distancia < menor)
         menor = maior_distancia;
      maior_distancia = -1;
   }
   printf("%d\n", menor);

   return 0;
}
```

Prefeito Tecnológico (Problema da Mochila)

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

const int maxn = 100;
int custos[maxn];
int votos[maxn];
int mtx[maxn][maxn];
int gastos[maxn][maxn];
int maior;

inline int max(int a, int b){
   maior = a<b ? b: a;
   return maior;
}
int main(){
   int t;
   int qtd_itens, caixa;

   scanf(" %d", &t);

   while(t--){
      scanf(" %d %d", &caixa,&qtd_itens);

      for (int i = 1; i <= qtd_itens; ++i){
         scanf(" %d %d", &custos[i], &votos[i]);
         mtx[i][0] = 0;
      }

      for(int i =0; i<=caixa; ++i)
         mtx[0][i] = 0;

      memset(gastos, 0, sizeof(gastos));

      for(int i = 1; i<=qtd_itens; ++i)
         for (int j = 1; j<=caixa; ++j)
            if(custos[i] > j){
               mtx[i][j] = mtx[i-1][j];
               gastos[i][j] = gastos[i-1][j];
            }
            else{
               if(mtx[i-1][j] < mtx[i-1][j - custos[i]] + votos[i]){
                  mtx[i][j] = mtx[i-1][j - custos[i]] + votos[i];
                  gastos[i][j] = gastos[i-1][j - custos[i]] + custos[i];
               }
               else{
                  mtx[i][j] = mtx[i-1][j];
                  gastos[i][j] = gastos[i-1][j];
               }
            }

      mtx[qtd_itens][caixa] ? printf("%d %d\n", mtx[qtd_itens][caixa], caixa-gastos[qtd_itens][caixa]) :
printf("NO FUNDS\n");
   }
   return 0;
}
```

Fatorial (Algoritmo do Troco)

```
#include<cstdio>

int main(){
    int n;
    int fatoriais[10];

    scanf(" %d", &n);

    fatoriais[0] = 1;
    for(int i = 1; i<=8; ++i)
        fatoriais[i] = fatoriais[i-1] * i;

    //for(int i = 0; i<8; ++i)
        //printf("%d ", fatoriais[i]);
    //printf("\n");

    int aux=0;
    for(int i = 8; i>=1;--i){
        if(n>=fatoriais[i]){
            aux += n/fatoriais[i];
            n %= fatoriais[i];
        }
    }

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

    return 0;
}
```

Troco (Subset sum – se há determinada soma em subconjunto)

```
#include<cstdio>
#include<algorithm>

inline int cmp(int a, int b){
   return a>b;
}

int main(){
   int v, m, solucao_atual;

   scanf(" %d %d", &v, &m);

   int moedas[m];
   int tabela[m+1][v+1];

   for(int i = 1; i<=m; ++i)
      scanf(" %d", &moedas[i]);

   std::sort(moedas+1, moedas+m+1, cmp); // Ordena do maior para o menor.

   for(int i = 0; i<=m; ++i)
      tabela[i][0] = 1;

   for (int j = 1; j<=v; ++j){
      tabela[0][j] = 0;
   }

   for(int i = 1; i<=m; ++i)
      for(int j = 1; j<=v; ++j){
         if(j - moedas[i] >= 0)
            if(moedas[i] - j == 0)
               tabela[i][j] = 1;
            else if(tabela[i-1][j])
               tabela[i][j] = 1;
            else if(tabela[i-1][j-moedas[i]])
               tabela[i][j] = 1;
            else
               tabela[i][j] = 0;
         else
            tabela[i][j] = 0;
      }

   printf(tabela[m][v] ? "S\n" : "N\n");

   return 0;
}
```

URI 1307 (Maior Divisor Comum - Elevar Números Inteiros - Inteiro para Binário)

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

using namespace std;

int gcd(unsigned int a, unsigned int b){
   return b == 0 ? a: gcd(b, a%b);
}

int apow(int n, int m){
   int r = 1;
   while(m){
      if (m & 1)
         r *= n;
      m >>=1;
      n *= n;
   }
   return r;
}

unsigned int to_bin(string s){
   int size_s = s.size();
   unsigned int c = 0;
   for(int i = size_s-1; i>=0; --i){
      if (s[i] == '1'){
         c += apow(2, (size_s-1)-i);
      }
   }
   return c;
}

int main(){
   int t, c=0;
   string s1;
   string s2;

   scanf(" %d", &t);

   while(t--){
      cin >> s1 >> s2;

      unsigned int a = to_bin(s1);
      int b = to_bin(s2);

      if (gcd(a, b) == 1)
         printf("Pair #%d: Love is not all you need!\n", ++c);
      else
         printf("Pair #%d: All you need is love!\n", ++c);
   }
   return 0;
}
```