./_0build.sh

```
#!/bin/bash
if [ -e "$1" ]
then
    rm -vf ./${1}.out
    g++ -g -std=c++11 $1 -o ${1}.out && ./${1}.out < test.txt
else
    echo "Need argument"
fi
```

./limits.cpp

```
#include <iostream>     // cout
#include <limits>       // numeric_limits

using namespace std;

int main () {
  cout << boolalpha; // print true or false instead of 1 or 0.
  //Maximun and Non-sign bits
  cout << numeric_limits<short>::max() << endl;
  cout << numeric_limits<short>::digits << endl;

  cout << numeric_limits<int>::max() << endl;
  cout << numeric_limits<int>::digits << endl;

  cout << numeric_limits<long>::max() << endl;
  cout << numeric_limits<long>::digits << endl;

  cout << numeric_limits<long long>::max() << endl;
  cout << numeric_limits<long long>::digits << endl;

  long long sum = 1;
  for (int i = 0; i < 35; i ++){
    cout << i << "\t\t" << sum << endl;
    sum = sum << 1;
  }
  /*
0               1                                       18              262144
1               2                                       19              524288
2               4                                       20              1048576
3               8                                       21              2097152
4               16                                      22              4194304
5               32                                      23              8388608
6               64                                      24              16777216
7               128                                     25              33554432
8               256                                     26              67108864
9               512                                     27              134217728
10              1024                                    28              268435456
11              2048                                    29              536870912
12              4096                                    30              1073741824
13              8192                                    31              2147483648
14              16384                                   32              4294967296
15              32768                                   33              8589934592
16              65536                                   34              17179869184
17              131072
  */
  return 0;
}
```

./basic.cpp

```cpp
#include <iostream>  // get, getline
#include <cstdio>    // printf, scanf
#include <algorithm> // sort
#include <string>    // string
#include <utility>   // pair
#include <list>      // list
#include <vector>    // vector
#include <map>       // map

#define FORIT(i,c) for (__typeof__((c).begin()) i = (c).begin(); i != (c).end(); i++)

using namespace std;

long long gcd(long long a, long long b)
{
    while(b) b ^= a ^= b ^= a %= b;
    return a;
}

int main(){
    // Input string
    string t;
    getline(cin, t);

    // Input char
    char c;
    cin.get(c);

    // GCD
    cout << gcd(991,3) << endl;

    // pair
    pair<int,string> p = make_pair(2,"hello world");
    cout << p.first << "|" << p.second << endl;

    // vector
    //O(1) : insertion and removal of elements at the end
    //O(n) : insertion and removal of elements at the beginning or in the middle
    vector<int> v; v.clear();
    v.push_back(1); v.push_back(2); v.push_back(3);
    int n = v.back(); v.pop_back(); cout << n << endl;  // 3

    // list
    //O(1) : insertion and removal of elements at the at the beginning or the end, or in the middle
    //O(n) : accessing
    list<int> l;
    l.push_back(0); l.push_back(1); l.push_front(2); l.push_front(3); //3201
    l.insert(++ ++ l.begin(), 9); //32901
    FORIT(i,l) cout << " " << *i;
    cout << endl;


    // map
    // use count() to find
    map<string,int> m;
    m["gg"] = 233; cout << m["gg"] << "|" << m["nosense"] << endl; //233|0
    cout << m.count("gg") << "|" << m.count("nosense") << endl; // m["nosense"] == 0
    m.erase("nosense"); cout << m.count("gg") << "|" << m.count("nosense") << endl;

    return 0;
}
```

./map-example.cpp

```cpp
#include <iostream>
#include <string>
#include <map>

using namespace std;

map<string, int> dictionary;
#define FORIT(i,c) for (__typeof__((c).begin()) i = (c).begin(); i != (c).end(); i++)

int main(){
    string t;
    dictionary.clear();
    while (cin >> t){
        if (t == "def"){
            cin >> t;
            int val; cin >> val;
            dictionary[t] = val;
        }else if (t == "calc"){
            bool unknown = false; bool method = true; //+ true - false;
            int result = 0;
            while (cin >> t){
                cout << t << " ";
                if (!unknown){
                    if (t == "+"){
                        method = true;
                    }else if (t == "-"){
                        method = false;
                    }else if (t == "="){
                        bool done = false;
                        // map iterator operation
                        FORIT(i,dictionary) if (result == i->second) {
                            cout << i->first << endl; done = true;
                            break;
                        }
                        if (!done) cout << "unknown" << endl;
                        break;
                    }else{
                        // map find() operation
                        if (!dictionary.count(t)){
                            unknown = true;
                        }else{
                            if (method){
                                result += dictionary[t];
                            }else{
                                result -= dictionary[t];
                            }
                        }
                    }
                    continue;
                }else{
                    if (t == "="){
                        cout << "unknown" << endl;
                        break;
                    }
                }
            }
        }else if (t == "clear"){
            dictionary.clear();
        }
    }
    return 0;
}
```

./top-sort.cpp

```cpp
#include <cstdio>
#include <iostream>
#include <string>
#include <vector>
#include <set>
#include <deque>
#include <algorithm>
#include <map>
#include <cstring>

using namespace std;

#define FOR(i,a,b) for (int i = (a); i < (b); i++)
#define FORIT(i,c) for (__typeof__((c).begin()) i = (c).begin(); i != (c).end(); i++)
#define FORITR(i,c) for (__typeof__((c).rbegin()) i = (c).rbegin(); i != (c).rend(); i++)
#define MAX 20

//normal
vector<int> adj[MAX][2];
int n, m;

// Topsort
vector<int> ts_list;
int ts_state[MAX];
int ts_dist[MAX];
bool topsort_loop;

void topsort_dfs(int current){
    if (ts_state[current] == 1) topsort_loop = true;
    if (ts_state[current]) return;
    ts_state[current] = 1;
    FORIT(i,adj[current][0]) topsort_dfs(*i);
    ts_state[current] = 2;
    ts_list.push_back(current);
}

void topsort(){
    topsort_loop = false;
    ts_list.clear();
    memset(ts_state, 0, sizeof(ts_state));
    FOR(i,0,n) topsort_dfs(i);

    reverse(ts_list.begin(), ts_list.end());

    //Print out Result
    cout << "topsort: ";
    FORIT(i,ts_list) cout << " " << *i + 1;
    cout << endl;
    cout << (topsort_loop ? "has loop":"no loop") << endl;
}
```

```
// DAG - Algorithm
void dag_short_paths(int b, int e){
    // find the shorted path between [b]egin and [e]nd
    memset(ts_state, 0, sizeof(ts_state));
    memset(ts_dist, 0, sizeof(ts_dist));

    // fill points with orders
    int ts_order[MAX];
    int c = 0;
    FORIT(i,ts_list) ts_order[*i] = c++;


    // find the starting point
    // In this example, all the path length is 1
    ts_state[ts_order[b]] = 1; // mark starting point valid.
    cout << "s:" << ts_order[b] << "\tt:" << ts_order[e] << endl;

    FOR(i,ts_order[b],ts_order[e]) FORIT(j,adj[ts_list[i]][0]) if (ts_state[ts_order[*j]]){
        int p = 1;
        if (ts_dist[ts_order[*j]] > ts_dist[i] + p) ts_dist[ts_order[*j]] = ts_dist[i] + p;
    }else{
        ts_state[ts_order[*j]] = 1;
        int p = 1;
        ts_dist[ts_order[*j]] = ts_dist[i] + p;
    }

    //print result
    cout << "Has path:" << ts_state[ts_order[e]] << endl;
    cout << "length  :" << ts_dist[ts_order[e]] << endl;

    cout << "DAG-SP: ";
    FOR(i,0,n) cout << " " << ts_dist[i];
}

// scc id from 0..N
int main(){
    cin >> n >> m;
    FOR(i,0,n){
        adj[i][0].clear();
        adj[i][1].clear();
    }

    FOR(i,0,m){
        int f,t;
        cin >> f >> t;
        f--;t--;
        adj[f][0].push_back(t);
        adj[t][1].push_back(f);
    }

    FOR(i,0,n){
        cout << i + 1 << " to:";
        FORIT(j,adj[i][0]) cout << *j + 1<< "\t";
        cout << "\tfrom:";
        FORIT(j,adj[i][1]) cout << *j + 1<< "\t";
        cout << endl;
    }

    //Topsot
    topsort();
    dag_short_paths(1,16);
    return 0;
}
```

./kosaraju.cpp

```cpp
#include <cstdio>
#include <iostream>
#include <string>
#include <vector>
#include <set>
#include <deque>
#include <algorithm>
#include <map>
#include <cstring>

using namespace std;

#define FOR(i,a,b) for (int i = (a); i < (b); i++)
#define FORIT(i,c) for (__typeof__((c).begin()) i = (c).begin(); i != (c).end(); i++)
#define MAX 1000000


//normal
vector<int> adj[MAX][2];
int n, m;

//kosaraju
// scc id from 0..N
int scc_n, scc_list[MAX];
vector<int> scc_stack;
set<int> scc_group[MAX];

void kosaraju_first_dfs(int node){
    if (scc_list[node] != -1) return;
    scc_list[node] = 0;
    FORIT(i,adj[node][0]) kosaraju_first_dfs(*i);
    scc_stack.push_back(node);
}

void kosaraju_second_dfs(int node, int scc_id){
    if (scc_list[node] != -1) return;
    scc_list[node] = scc_id;
    FORIT(i,adj[node][1]) kosaraju_second_dfs(*i, scc_id);
}

void kosaraju(){
    memset(scc_list, -1, sizeof(scc_list));
    scc_stack.clear();
    FOR(i,0,n) kosaraju_first_dfs(i);

    scc_n = 0;
    memset(scc_list, -1, sizeof(scc_list));
    reverse(scc_stack.begin(),scc_stack.end());
    FORIT(i,scc_stack) if (scc_list[*i] == -1) kosaraju_second_dfs(*i,scc_n ++);

    // TEST PRINT OUT
    //FOR(i,0,n) cout << i+1 << " : SCC_ID" << scc_list[i] << endl;
}
```

```
//Dynamic Programming
int dp_length[MAX];
int dp_result[MAX];

void dp_fill_distance_dfs(int node, set<int> next, int distance){
    if (distance > dp_length[node]) dp_length[node] = distance;
    next.erase(node);
    FORIT(j,adj[node][0]) if (next.count(*j)) dp_fill_distance_dfs(*j,next,distance+1);
}

int main(){
    cin >> n >> m;
    FOR(i,0,n){
        adj[i][0].clear();
        adj[i][1].clear();
    }

    FOR(i,0,m){
        int f,t;
        cin >> f >> t;
        f--;t--;
        adj[f][0].push_back(t);
        adj[t][1].push_back(f);
    }

/*
    FOR(i,0,n){
        cout << i + 1 << " to:";
        FORIT(j,adj[i][0]) cout << *j + 1<< "\t";
        cout << "\tfrom:";
        FORIT(j,adj[i][1]) cout << *j + 1<< "\t";
        cout << endl;
    }
*/
    // Finding Strongly Connected Components
    kosaraju();

    // Clean up a little bit scc
    // scc_group[0] stores all the isolated nodes.
    FOR(i,0,scc_n){
        scc_group[i].clear();
    }
    FOR(i,0,n){
        scc_group[scc_list[i]].insert(i);
    }

    //Print out scc group
/*
    FOR(i,0,scc_n){
        cout << "group id:" << i << "\t";
        FORIT(j,scc_group[i]) cout << *j + 1 << "\t";
        cout << endl;
    }
*/
```

```
    // ------------ Dynamic Programming -------------
    //memset(dp_length, -1 ,sizeof(dp_length));
    memset(dp_result, 0 ,sizeof(dp_result));
    // Finding Result "longest path"
    int max_length = 0;
    FOR(i, 0, scc_n){
        FORIT(p,scc_group[i]){
            //loose
            FORIT(j,scc_group[i]) dp_length[*j] = -1;
            //DFS from the node *p
            dp_fill_distance_dfs(*p,scc_group[i],1);
            int length = 0;
            //From outside of the SCC
            FORIT(j,adj[*p][1]) if((i != scc_list[*j]) && (dp_result[*j] > length)) length =
dp_result[*j];
            //From inside the SCC
            FORIT(j,scc_group[i]) if (dp_result[*j] < (length + dp_length[*j])) {
                dp_result[*j] = (length + dp_length[*j]);
                if (dp_result[*j] > max_length) {
                    max_length = dp_result[*j];
                }
            }

        }
    }

    cout << max_length << endl;
    return 0;
}
```