//Strongly Connected Component (Tarjan)

//Complexity : O(V+E)

vector<int>G[30], SCC;

int dfs\_num[30], dfs\_low[30], dfsCounter, SCC\_no = 1;

bitset<30>visited;

void tarjanSSC(int u) {

//Stack, here, it is implemented as vector instead

SCC.push\_back(u);

//Marking node u as visited

//visited[u] marks if the node u is usable in a SCC and not used on other SCC

//if visited[u] is false, then it is used in other SCC

visited[u] = 1;

dfs\_num[u] = dfs\_low[u] = ++dfsCounter;

//for all Strongly Connected Component (directed graph), dfs\_low[u] is same

for(int i = 0; i < G[u].size(); i++) {

int v = G[u][i];

//if it is not visited yet, backtrack it

if(dfs\_num[v] == 0) {

tarjanSSC(v);

}

//if node v (visited[v]) is not visited, we can use it to minimize the dfs\_low[u] value from dfs\_low[v]

if(visited[v])

dfs\_low[u] = min(dfs\_low[u], dfs\_low[v]);

}

//in a SCC the first node of the SCC, node u is the first node in a SCC if dfs\_low[u] == dfs\_low[v]

//as we implementing stack like data structure, the nodes from top to u are on the same SCC

if(dfs\_low[u] == dfs\_num[u]) {

bool first = 1;

printf("SCC %d\n", SCC\_no++);

while(1) {

int v = SCC.back();

SCC.pop\_back();

//node v is used, so marking it as false, so that the ancestor nodes

//doesn't use this node to update it's value

visited[v] = 0;

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

if(u == v)

break;

}

printf("\n");

}

}

int main()

{

//tarjanSCC main code

memset(dfs\_num, 0, sizeof(dfs\_num));

dfsCounter = 0;

visited.reset();

for(int i = 1; i < indx; i++) {

if(dfs\_num[i] == 0)

tarjanSSC(i);

}