connectedSet.cpp

```cpp
//Implementation of Kosaraju's algorithm to print all SCCs
#include <iostream>
#include <list>
#include <stack>
using namespace std;

class Graph {
    int V;
    list<int> *adj;

    void fillOrder(int v, bool visited[], stack<int> &Stack);

    void DFSUtil(int v, bool visited[]);

public:
    Graph(int V);
    void addEdge(int v, int w);

    void printSCCs();

    Graph getTranspose();
};

Graph::Graph(int V){
    this->V = V;
    adj = new list<int>[V];
}

void Graph::DFSUtil(int v, bool visited[]){
    visited[v] = true;
    cout << v << " ";

    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFSUtil(*i, visited);
}

Graph Graph::getTranspose() {
    Graph g(V);
    for (int v = 0; v < V; v++) {
        list<int>::iterator i;
        for(i = adj[v].begin(); i != adj[v].end(); ++i) {
            g.adj[*i].push_back(v);
        }
    }
    return g;
}

void Graph::addEdge(int v, int w) {
    adj[v].push_back(w);
}

void Graph::fillOrder(int v, bool visited[], stack<int> &Stack) {
    visited[v] = true;

    list<int>::iterator i;
    for(i = adj[v].begin(); i != adj[v].end(); ++i)
        if(!visited[*i])
            fillOrder(*i, visited, Stack);
```

```
    Stack.push(v);
}

void Graph::printSCCs() {
    stack<int> Stack;
    bool *visited = new bool[V];

    for(int i = 0; i < V; i++)
        visited[i] = false;

    for(int i = 0; i < V; i++)
        if(!visited[i])
            fillOrder(i, visited, Stack);

    Graph gr = getTranspose();

    for(int i = 0; i < V; i++)
        visited[i] = false;

    while (!Stack.empty()) {

        int v = Stack.top();
        Stack.pop();

        if (!visited[v]) {
            gr.DFSUtil(v, visited);
            cout << endl;
        }
    }
}

int main() {
    Graph g(5);
    g.addEdge(1, 0);
    g.addEdge(0, 2);
    g.addEdge(2, 1);
    g.addEdge(0, 3);
    g.addEdge(3, 4);

    cout << "Following are strongly connected components in "
            "given graph \n";
    g.printSCCs();

    return 0;
}
```