Задача.

Рассмотрим ориентированный граф, у которого каждая вершина имеет только одно исходящее ребро. Такой граф может быть построен на основе структур Node типа узла однонаправленного списка. Т.е. для каждой вершины может существовать только одна последующая вершина (указатель next). Будем называть такое отношение вершин "родитель –> потомок" по направлению ребра this -> next. Итак, вершина может иметь не более одного потока, но может быть непосредственным потомком нескольких других вершин. Назовем рангом вершины количество ее родителей (количество входящих ребер графа). Вершины ранга 0 назовем стартовыми. Граф задается массивом указателей на его стартовые вершины, а доступ к остальным вершинам осуществляется по указателям next. Также граф может иметь (ориентированные) циклы.

Вам выдается заготовка кода, которая содержит описание графа и определения некоторых его функций. В частности, это функции LoadForTestN, создающие граф некоторой конфигурации для тестов. Функции даны в исходных кодах. Для устранения возможных проблем с русскими кодировками комментарии в заготовках даны на английском языке.

Граф имеет функцию для распечатки его состояния, однако эта функция реализована очень примитивно и не работает для графов, содержащих циклы. В этом случае она зацикливается и ее не следует применять для циклических графов.

Компиляция программы выполняется командами g++ graph-test.cpp graph.cpp. Выполните компиляцию и запустите a.out чтобы убедиться, что заготовка компилируется и работает.

При решении задачи вы можете вносить любые дополнительные поля и методы в классы Node и Graph. Нельзя только изменять права доступа (private/public) для уже имеющихся полей этих структур. При этом надо понимать, что после изменений уже вы несете ответственность за работоспособность программы. На эффективность программы особых требований не накладывается. Однако утечки памяти считаются ошибками.

Задание. Для данного графа надо определить количество связных компонент графа и распечатать id стартовых вершин, относящихся к каждой связной компоненте. Вершины принадлежат одной связной компоненте, если они имееют одного и того же потомка (возможно, не сразу, а через несколько шагов по ребрам).

К задаче прикладываются несколько тестов. Для проверки надо оставить одну из функций LoadForTestN (N = 1, 2, 3, 4, 5), а остальные закомментировать. Программа должна работать на всех тестах. Весь каталог с файлами вашей реализации надо положить в один zip архив и отправить на почту своему преподавателю. Вывод результатов нужно выполнять в отдельный файл, чтобы проверяющий смог увидеть как ваша программа отработала на вашем компьютере. Так как ваша программа будет проверяться независимо от вас, то следует сделать выдачу интуитивно понятной, либо отдельно пояснить что в ней есть что. В заголовке письма нужно написать Nгруппы - фамилия - экзамен (скажем, 207-Садовничий-экзамен) имя архива также желательно назвать аналогичным образом.