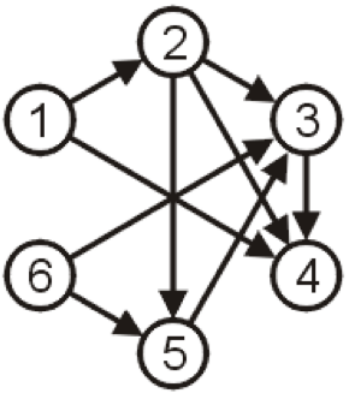
COEN 244 Winter 2017 Session W Project Specification

Problem statement

A directed graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are directed from one vertex to another. In contrast, a graph where the edges are bidirectional is called an undirected graph. A graph can be formally define as *G=(N,E)* consisting of the set *N* of nodes and the set *E* of edges, which are ordered pairs of elements of *N*. It is further assumed in this project specification that any graph is finite with no directed or undirected cycles. An example is given below for directed graph.

$$V = \{1,2,3,4,5,6\}$$
$$E = \{(1,2),(1,4),(2,3),(2,4),(2,5),(3,4),(5,3),(6,3),(6,5)\}$$



A graph has many useful applications in the real world, such as scheduling of tasks; network topology; family tree; data compression; literature citation, social connections and so on.

This project requires you to develop object oriented programs of a graph that can achieve the following functions.

1. A graph can be empty with no vertex or edge.
2. A graph can be either a directed graph or an undirected graph.
3. A graph can be added in vertices and edges.
4. A vertex of a graph can contain values – in theory, the values can be of any type.
5. A graph can be displayed by listing all the possible paths, each linking vertices.
6. A graph can be queried by given a starting vertex, listing the path this vertex leads.
7. A graph can be queried by given an edge, if this edge exists in the graph
8. A graph can be queried if a value is contained by any of its vertex.

Deliverables

There are 2 parts of deliverables of this project that needs to fulfill the above functional requirements.

Part 1: Programing Code in one Zip file in the form of **[group_member1_SID]-[group_member2_SID]-code.zip. [Totally 40 points]**

1. Design and program necessary C++ Classes with data members and member functions for above functions.
2. For each Class designed, provide default constructor, copy constructor, and constructor with arguments.
3. A Driver application to test the graph and demonstrate its functions.
4. Apply at least three of the following techniques (any 3): inheritance; polymorphism; operator overloading; template; exception handling.

Part 2: Report in one PDF File, following IEEE paper format. Attached to this specification for downloading. **[group_member1_SID]-[group_member2_SID]-report.pdf [Totally 10 points]**

5. Design Description: (a) Provide a Class Diagram that describes, in detail, the classes you employ, and the relationships between them (following UML conventions); (b) Describe, using pseudo-code, at least 2 non-trivial methods (member function) in your program; (c) describe your usages of 3 of techniques from inheritance, polymorphism, operator overloading, template and exception handling.

6. Testing Results: Utilize a black-box testing methodology of your program, with sufficient test cases, to support the hypothesis that your program is bug-free (does not crash in response to valid inputs), correct (generates the right output, in response to valid input) and reasonably robust (does not crash in response to – at least reasonable – invalid inputs). Necessary screenshots should be added. You can also consider providing a link to the video demo of your program.

Important Date

1. **By Jan 16th,** send information to TA your group information, without sending this information, a penalty of 1 point will be deducted.
2. **By Feb 25th, demonstrate the up-to-date version of your program code to TA during the tutorial time in the lab. Submit to the moodle site the up-to-date report for reviews and comments on your progress**. Note that this demo and submission is not graded by techniques but for providing comments to your status of the project. However, no demo or no report will result in 5 penalty points deducted. The late submission police applies.
3. **By April 10th, final project program code and report should be submitted to the moodle site for final marking.**
4. Randomly selected groups or volunteering groups will present the critical features of their design and programs at the lecture time **on Week 14**. Bonus points up to 5 points will be awarded to each team member for the presentation. If selected but no presentation will result in 5 penalty points deducted.