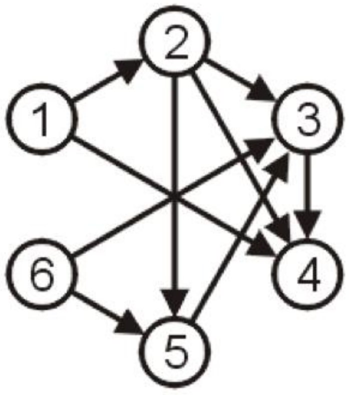
COEN 244 Winter 2022 Project Specification

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 an object-oriented application of a graph. Example applications can be social connection network, equipment connections, course sequences, family trees and so on. The graph application should achieve the following base **function:**

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.

Group Requirements

A group of maximum two members.

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. **[group_member1_SID]-[group_member2_SID]-report.pdf [Totally 10 points]**
https://www.ieee.org/conferences/publishing/templates.html

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 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 Dates

Timeline	Task
Week 3 tutorial time	Send information to tutorial session TA your group information. Missing the submission has 1 point deduction.
Week 9 tutorial time	Demonstrate to tutorial session TA at least 5 out of 8 graph functions and two techniques implemented by running the driver application. Missing the demo to TA has 5 point deduction
Week 13 tutorial time	Demonstrate the whole application program to tutorial session TA by running the driver application. Missing the demo to TA has 5 points deduction.

The last lecture time	Randomly selected groups or volunteering groups will present the critical features of their design and programs at the last lecture time by sharing zoom screen. Bonus points up to 3 points will be awarded to each team member for the presentation. If selected but no presentation will result in 5 points deduction.
April 19, 11:59	Final project program code and report should be submitted to the Moodle site for grading.

Appendix

Week 9 Tutorial Time Demo Checklist

Five out of eight graph 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.
[9] Show one C++ class defined with default constructor, copy constructor, and constructor with arguments.
[10] A Driver application to test the graph and demonstrate its functions.

Week 13 Tutorial Time Demo Checklist
The rest three graph 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.
[9] Demonstrate at least three of the following techniques (any 3): inheritance; polymorphism; operator overloading; template; exception handling.

Lecture Time Presentation Checklist
[1] Present with a Class Diagram that describes, in detail, the classes you develop, and the relationships between them (following UML conventions)
[2] Present at least 2 non-trivial methods (member function) in your classes developed;
[3] Present the usage of 3 kinds of techniques from inheritance, polymorphism, operator overloading, template and exception handling.