UAH Electrical and Computer Engineering Department
Project06 (4 points) Version 1.0

Submit Your Solution to the Canvas Dropbox by
5:00pm on Friday, April 19, 2019

Important Notes

Start early to give yourself an opportunity to ask questions!!!

Waiting to the last minute to begin the assignment is not recommended and will not get you an extension on the submission deadline.

Debugging issues are far easier to resolve in person when we can see exactly what you and your code are doing.

All CPE 212 projects are *automatically graded* in order to provide you timely feedback. So, it is critical that you follow all directions for the preparation and submission of your projects for grading. Failure to follow the directions may result in zero credit (0 points).

Hint

Match the output of your program to that of the sample solution!!!
Directions for running the sample solution appear below.
Directions for running the preview script appear on the following pages.

Project06 Description

Complete the provided partial C++ program that will implement a ***Graph ADT Class*** in which an ***Adjacency List*** as the internal representation of the graph structure.

The lecture notes and textbook provide the source code for basic graph methods and the DFS and BFS algorithms but you will need to modify this code to complete this assignment.

Required modifications include
– conversion to use an adjacency list representation
– addition of the specified exception detection/handling
– conversion to use the ***stack*** and ***queue*** containers from the Standard Template Library.

Additional details may be found on pages 4-5 of the handout (see ***Project06 Hints***).

Running the Sample Solution on blackhawk or the laboratory Linux machines

The best description of what your code must do is the Sample Solution for the project.

Run the sample solution by typing the following at **blackhawk** terminal window command prompt where `inputfilename` is the name of one of the provided input files (for example, ***p06input1.txt***).

```
/home/work/cpe212/project06/p06    inputfilename
```

Your current working directory must contain the input files for this to work.

Unzipping Sample Input Files on blackhawk

Use the Firefox browser to access Canvas and download **main.cpp** and the sample input files into your **Project06** directory. At terminal window prompt, use the unzip utility to uncompress the files. For example, to unzip the files into your current directory:

```
unzip  Project06_Materials.zip
```

Since this project is worth four points, you have been given four input files to test your program.

Running the Preview Script on blackhawk

Preconditions for running the Preview script:

(1) ***Your current working directory must contain both your project06 executable and all sample input files BEFORE you execute the preview script!!!***

(2) Linux is case sensitive – be sure the name of your executable is ***project06***

Run the preview script by typing the following in a **blackhawk** terminal window command prompt

/home/work/cpe212data/project06/preview06.bash

This script will run both the Sample Solution AND your project06 executable program on the complete set of input files, and it compares the outputs of the two programs line by line to identify errors in your program's outputs. Make sure that the output of your program exactly matches the output of the Sample Solution.

Be sure to fix any memory leaks identified by the preview script to receive full credit.

Project06 Files Provided – DO NOT MODIFY OR SUBMIT THESE FILES **

graph.h – declares the ***Graph ADT*** class and describes its member functions

main.cpp – includes the **main()** function test driver for the ***Graph*** class.

makefile – includes all commands required to compile and link your project on ***blackhawk***

Do not modify or submit any of the provided files named above** !!!

Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.
All output to the monitor *(stdout) will be performed by the code provided.*

Project06 FILES YOU MUST WRITE AND SUBMIT **

graph.cpp – add your code here to implement the member function of the Graph class.

The code in graph.h must not include any INPUT or OUTPUT statements.

PRINT functions have been included in graph.h for debugging purposes

Use of the Standard Template Library containers or container adapters is not allowed.

===> Submit only the SOURCE CODE in graph.cpp for grading. <===**

****Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.**

How to get started on this assignment?

See graph.h for descriptions of Graph ADT.

Once again, I recommend creating an Empty Framework of function definitions in graph.cpp that compiles first – before adding any code to any function.

Address the critical operations within the Graph ADT [constructor, AddVertex(), AddEdge()] since these critical items must be functional in order to test other operations.

Make sure these critical operations work before adding code for any non-critical operations.

Program Compilation Instructions

This project consists of two C++ files provided by the instructor, along with a file named **makefile** to help you compile your program. So, for this assignment, all you must do to compile the program is to use the following command at the Linux command line

make

which will create an executable named **project06** from the provided files.

If your program compiled successfully, you may then type

./project06 NameOfInputFile

to execute your program assuming that the input file is located in the same directory with the executable.

Submission Instructions

Submit only the file graph.cpp as an attachment to your Canvas dropbox submission.

Submissions that are incomplete will receive zero credit (0 points).

Submissions that do not compile will receive zero credit (0 points).

Submissions by email will receive zero credit (0 points).

Submissions that are late will receive zero credit (0 points).

Project06 Include File Constraints **

All allowed include files within graph.cpp appear below:

#include "graph.h"

#include <stack>

#include <queue>

Use of the Standard Template Library containers or container adapters (other than ***stack*** and ***queue***) is not allowed. Includes for stack and queue appear within **graph.h**

**** Failure to follow these directions will result in zero (0) credit on this assignment.**

Project06 Hints

You will be better prepared for the Final Exam if you can write the **Graph ADT** code on your own rather than just copying code from the textbook or lecture notes since the written exams are CLOSED BOOK and CLOSED NOTES.

When adapting the code provided during lecture, you will need to make modifications to convert the code to work with the **Adjacency List** representation of a graph. The **VertexExists** and **EdgeExists** methods take the place of the **IndexIs** method since there is **NO ARRAY IN THIS PROJECT**!!!

For help with the **stack** and **queue** containers from the Standard Template Library, see the lecture notes.

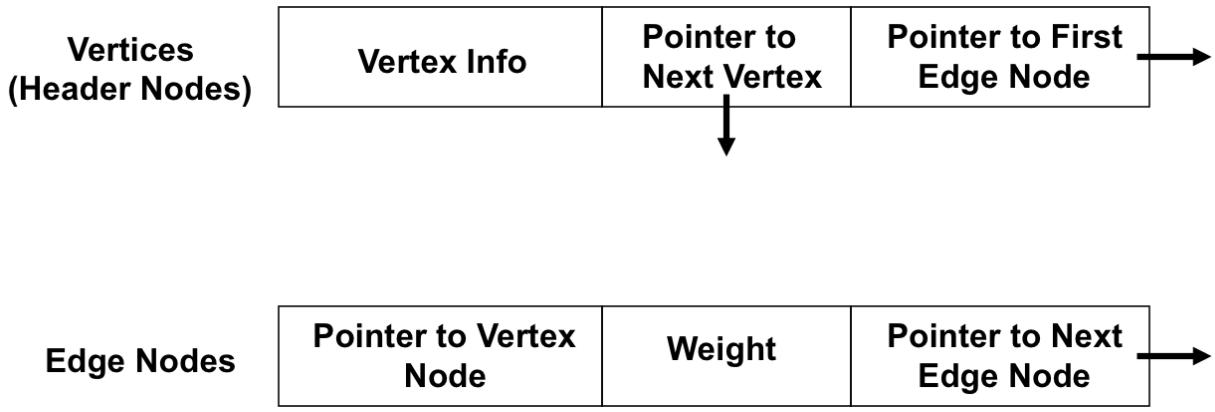
Project06 Adjacency List Representation

VertexNode and **EdgeNode** are structs used to represent vertices and edges of the graph in the adjacency list representation.

In the Adjacency List implementation, there is **NO ARRAY**!!!

Within the constructor, you will need to initialize the pointer to the list of vertices to NULL.

Detailed descriptions of the **VertexNode** and **EdgeNode** are structs may be found in the **graph.h** file.



Reminders:

- The numbers in the left-hand column do not exist (they are not array index values). These values are used to indicate the vertex referred to by an edge node. (They are simplified stand-ins for memory addresses)

  For example:

  The **5*** means the address of the vertex node representing Houston.