PAssignment Objective: Build skills on C class creation and integration while implementing a **Circular Queue** ADT and extending the **Graph** ADT to do some graph processing.

Requirements:

* Note, this uses the p1.cpp and p1.h files that were used in p9a.
* Create a new Circular Integer List, **cIntList**, derived from the P3 assignment. Create cil.cpp and cil.h from your p3.cpp and p3.h, converting the code from a list of strings to a list of integers. Note, the only cIntList functionality required is as follows: constructor/destructor, add(), deleteFirst(), printIt(), clear(), and count().
* Create integer queue, **iQ,** ADT with the following variables and methods. This shall be done in iq.cpp and iq.h. And it shall use the cIntList
  + Private area:
    - cIntList \*theList;
  + Public area:
    - iQ(int n = 100); // the constructor
    - ~iQ(); // the destructor; deletes the list structure
    - bool enq(int v); // enqueues the value v into the iQ
    - bool deq(int &v); // dequeues the oldest value, returns that value in v; returns true if queue had something; returns false if the iQ was empty; also sets v to -1 for returning from an empty iQ
    - void clear(); // removes all items in the iQ, resulting in the iQ being empty
    - int count() const; // returns the number of items in the iQ
    - void printiIt() const; // prints the contents of the iQ
* Copy the p9a.cpp to p9b.cpp.
* Copy the p9a.h to p9b.h.
* Within the p9b files, add the following additional methods and variables:
  + Public method:
    - void bfPrint(int label) const; // this is a print traversal of the graph, starting at the given node. The output must match the p9bCorrectOutput.txt format. The traversal is a breadth-first traversal.
    - bool isPath(int ulabel, inv vlabel) const; // this is a breadth-first traversal that DOES NOT PRINT. It determines whether there is a path from the first node, ulabel, to the second node, vlabel. Returns true if the path exists; returns false if either label is not a valid node or there is no path from the first label to the second.
    - void printPaths() const; // This function prints the status of whether there is a path from each node, to each other node. The output format must match that of the p9bCorrectOutput.txt file.
  + Private variable and function:
    - iQ \*q; // pointer to an iQ ADT. Note, the iQ will get allocated in the Graph constructor; and get destroyed in the Graph destructor.
    - int vidToLabel(int vid) const; // Given a vertex id, it returns the label associated with the vertex. If the vid is not valid, it returns -1. Whenever a conversion from vid to label is needed, use this function; do not directly access the list; only this function uses readAt on the list.
  + NOTE: p9b.h will need to include the iq.h file. But no other file should include the iq.h file directly. Note, the user of the GRAPH ADT has no knowledge of the implementation, hence has no need to know that an iQ is being used.
* You must not use any other data structure, whether built-in or otherwise.
* Demonstrate that the iQ ADT works:
  + Compile your program as follows:

g++ p3.cpp iq.cpp iqm.cpp -o iq

* + Run your program as follows:

iq > iqOutput.txt

* + Compare your output file, iqOutput.txt, to the posted iqCorrectOutput.txt file
* Demonstrate that the graph structure works:
  + Compile your program as follows:

g++ p1.cpp cil.cpp iq.cpp p9b.cpp p9bm.cpp -o p9b

* + Run your program as follows:

p9b < p9bInput.txt > p9bOutput.txt

* + Compare your output file, p9bOutput.txt, to the posted p9bCorrectOutput.txt file
* Deliverables:
  + Into D2L put a zip file containing:
    - A cil.h file and cil.cpp file for your cIntList ADT
    - An iq.h file and iq.cpp file for your iQ ADT
    - An iqOutput.txt file with your output from testing the iQ
    - A p9b.h file p9b.cpp for your updated graph ADT
    - A p9bOutput.txt text file with your output
    - DO NOT CHANGE THE NAMES OF THE FILES
    - DO NOT put a project into D2L
  + Turned into class: a hardcopy of the files above.