Assignment 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:

* Create the **queue** ADT with the following variables and methods. This shall be done in queue.cpp and queue.h. And it shall use the Circular KeyValue List ADT.
  + Private area:
    - cKeyValueList \*theList;
  + Public area:
    - queue(int n = 100); // the constructor
    - ~queue(); // the destructor; deletes the list structure
    - bool enq(int v); // enqueues the value v into the queue
    - bool deq(int &v); // dequeues the oldest value, returns that value in v; returns true if queue had something; returns false if the queue was empty; also sets v to -1 for returning from an empty queue
    - void clear(); // removes all items in the queue, resulting in the queue being empty
    - int count() const; // returns the number of items in the queue
    - void printiIt() const; // prints the contents of the queue
* Add to graph.h and graph.cpp the following additional member functions and variables:
  + Public function:
    - void printAllPaths(); // For every vertex, it prints whether a path exists to each vertex. CLRS says, for this, that a vertex has a path to itself. It is possible for the vertex to have a self-loop. If so, it shall print that there is a self-loop. See the correct output file for how the printing shall look.
  + Private variables:
    - queue \*q; // pointer to an queue ADT. Note, the queue will get allocated in the dGraph constructor; and get destroyed in the dGraph destructor.
    - const int INFINITE = 1000000;
    - const int vidNIL = -1;
  + Private functions:
    - void printPath(int uVid, int vVid) const; \*q; // recursive causes the path from uVid to vVid to be printed.
    - bfs(int uVid); // performs the BFS algorithm as described in class and in the CLRS text, using the vertex attributes that were defined in the prior assignment.
  + NOTE: graph.h will need to include the queue.h file. But no other file should include the queue.h file directly. Note that the graph ADT user has no knowledge of the implementation and hence has no need to know that a queue is being used.
* You must not use any other data structure, whether built-in or otherwise.
* Demonstrate that the **queue** ADT works:
  + Add the appropriate Makefile changes to create “queue” using the cList and queue ADTs, and the provided queueMain.cpp to build the executable file called queue.exe (or just queue).
  + Run your program as follows:

queue > queueOutput.txt

* + Compare your output file, queueOutput.txt, to the posted queueCorrectOutput.txt file
* Demonstrate that the graph structure works:
  + Modify the Makefile to also build “graph” with the cList.o and queue.o object files, and the provided graphBFSmain.cpp, taking care to account for the file dependencies.
  + Note that the “all:” rule should cause both queue and graph to be built.
  + Run your program as follows:

graph < graphInput.txt > graphBFSoutput.txt

* + Compare your output graphBFSoutput.txt file to the posted graphBFScorrectOutput.txt file.
* Deliverables:
  + Into D2L put a zip file containing: cList.h, cList.cpp, queue.h, queue.cpp, queueOutput.txt, graph.h, graph.cpp, graphBFSoutput.txt, and Makefile.
    - DO NOT CHANGE THE NAMES OF THE FILES
    - DO NOT put a project into D2L
  + Turned into class: a hardcopy of the files above, except for Makefile, in that order.