Assignment Objective: Build skills on C class creation and integration while implementing a **Max Priority Queue** with a **Heap.**

Requirements:

* Create a MAX priority queue class called **PQ** for integers. It will have the following members
  + Our standard KEY\_VALUE type definition
  + Private members:
    - Data:
      * An array “keyValues[]” to hold the KEY\_VALUE types
      * An integer “qCount” to record the number of entries in the PQ
      * An integer pqSize to record the size of the array.
    - Methods:
      * int parent(int index) const; // returns the parent’s index for the given index; returns 0 if index is 0
      * int left(int index) const; // returns the index of the left child of the given index
      * int right(int index) const; // returns the index of the right child of the given index
      * void printIt(int ind, int count) const; // prints the tree, level by level
      * void swap(KEY\_VALUE \*x, KEY\_VALUE \*y); // swaps two integers with each other
      * void bubbleUp(int index); // performs the heap bubble-up operation
      * void heapify(int i); // performs the heap heapify operation
  + Public members:
    - constructor PQ(int n) that causes the object to be initialized with an array of size n, defaulting to 100
    - destructor ~PQ() that deletes the array.
    - bool enq(KEY\_VALUE kv) – Enqueues the key-value into the PQ; returns true for success; returns false if the PQ was already full.
    - bool deq(KEY\_VALUE &kv) – returns true if the PQ is not empty; also removes and returns the key-value of the maximum kv in the PQ. Returns false if the PQ is empty.
    - void printIt() const – results in the keys of the queue being printed in breadth-first order; one line for each level of the tree, such as:

Line 1: (14,111)

Line 2: (22,222) (18,-77)

* + - void clear() – removes all the keys from the PQ, making it empty.
    - int count() const – returns the number of keys in the PQ
* You must not use any other data structure, whether built-in or otherwise.
* Make an appropriate Makefile
* Demonstrate that the PQ data structure works:
  + Run your program as follows:

make

./PQ > PQoutput.txt

* + Compare your output file, PQoutput.txt, to the posted PQcorrectOutput.txt file
* Deliverables:
  + Into D2L put a zip file containing PQ.h, PQ.cpp, PQoutput.txt, and Makefile.
    - DO NOT put a project into D2L
  + Submit in class a hardcopy of your PQ.h, PQ.cpp, and PQoutput.txt files.