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

Requirements:

* Create a class called i**PQ** for integers. It will have the following members
  + Private members:
    - Data:
      * An array “values[]” to hold the integers
      * An integer “qCount” to record the number of entries in the PQ
      * An integer qCapacity 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. HINT: consider using the log function (don’t forget to include <cmath>)
      * void swap(int \*x, int \*y); // swaps two integers with each other
      * void bubbleUp(int index); // performs the heap bubble-up operation
      * void heapify(int index); // performs the heap heapify operation
  + Public members:
    - constructor iPQ(int n) that causes the object to be initialized with an array of size n, defaulting to 100
    - destructor ~iPQ() that deletes the array.
    - bool enq(int v) – Enqueues v into the iPQ; returns true for success; returns false if the PQ was already full.
    - bool deq(int &v) – returns true if the iPQ is not empty; also removes and returns the maximum value in the PQ. Returns false if the iPQ is empty.
    - void printIt() const – results in the values of the queue being printed in breadth first order; one line for each level of the tree, such as:

Level[0]: Texas

Level[1]: Oklahoma Tennessee

Level[2]: Miami Newton Orlando Portland

Level[3]: Dallas

* + - void clear() – removes all entries from the iPQ, making it empty.
    - int count() const – returns the number of entries in the iPQ
* Deliverables:
* For P6a, implement the highlighted; use a simple printIt() to print the array in the order of the array.
* For P6b, turn in with the all features implemented as specified.
* Demonstrate that the iPQ data structure works:
  + Run your program as follows:

g++ p6.cpp p6m.cpp -o p6

./p6 > p6output.txt

* + Compare your output file, p6output.txt, to the posted p6correctOutput.txt file
  + Into D2L put a zip file containing:
    - A p6.h file for your iPQ ADT
    - A p6.cpp file for your iPQ ADT implementation
    - A p6output.txt text file with your output
    - DO NOT put a project into D2L
  + Turned into class, a hardcopy of your p6.h, p6.cpp, and p6output.txt files in the order given above.