Assignment Objective: Build skills on C class creation while implementing a list of integers.

Requirements:

* Into two files, list.h and list.cpp, create a class called “list” with the following members
  + Private members:
    - int \*keys // A pointer to the array into which the list keys will be stored.
    - int listCapacity; // memorialize the capacity of the list
    - int listCount; An integer to record the number of entries in the list
  + Public members:
    - constructor list(int capacity) that causes the array to be sized at “listCapacity” entries, with a default capacity of 100
    - destructor ~list() that deletes the dynamically allocated array “keys”
    - bool insert(int key) – inserts the key at the beginning of the list; all other entries shift right. Returns true if key was inserted; otherwise, it returns false.
    - bool add(int key) – inserts the key at the end of the list. Returns true if the key was inserted; otherwise, it returns false.
    - bool insertAt(int index, int key) – inserts the key at the index “index”; keys at that position need to shift right. Returns true if the key is inserted. Returns false if the list was full or if the index was greater than “listCount”.
    - bool deleteAt(int index, int &key) – deletes the key at the given index; shifts the entries right of that index to the left. If the index is within the range of the list, it sets “key” to the key at the index and returns true. Otherwise, it does not change “key” and returns false.
    - bool deleteFirst(int &key) – If the list is not empty, the function deletes the value at the first of the list, updating where first indexes (no copying is required here). The value deleted is returned via the referenced parameter and the function returns true. If the list was empty, the referenced value does not change value and the function returns false.
    - bool deleteLast(int &key) – If the list is not empty, the function deletes the value at the last of the list, updating where last indexes (no copying is required here). The value deleted is returned via the referenced parameter and the function returns true. If the list was empty, the referenced value does not change value and the function returns false.
    - bool readAt(int index, int &key) const – Same as deleteAt(), save that it does not delete the entry; it just returns the appropriate key.
    - void clear() – causes the list to be emptied
    - void printIt(int limit) const – causes limit items of the list to be printed, one key per line; for each line, print the index and the key at that index. If limit < 0 or limit > listCount, then print the whole list.
    - void printItBackwards(int limit) const – same as printIt() but prints the list from the “bottom” going “up”.
    - int getIndex(int key) const – returns the first position at which the key was found; otherwise returns -1.
    - int getCapacity() const – returns the capacity of the list
    - int getCount() const – returns the number of entries in the list
  + Note: All member functions that can be declared const should be declared as const.
* Create a Makefile per classroom instructions that makes each .o file and the final program, named list. This shall use the provided main() file.
* Demonstrate the correctness of the implementation by doing the following, using the listMain.cpp that is given to you on D2L:
  + Compile the code into the executable file list.exe (or list)
    - make list
  + Execute the code and inspect the results; if they are what you expect from your understanding of listMain.cpp, proceed to the submission; otherwise, inspect and fix your code as necessary and repeat these steps.
    - The correct output is shown in the D2L file listCorrectOutput.txt
* **Submission process:**
  + Create the file listOutput.txt using the following command in your environment:
    - ./list > listOutput.txt
    - cat listOutput.txt (this step ensures that you created the file correctly)
  + **Printing:**
    - Print a combined listing of only these files, in the following order: list.h, list.cpp, and listOutput.txt.
    - NO screenshots will be accepted.
    - Bring the listing to class when due.
  + **D2L submission:**
    - Submit a zip file to D2L with the Makefile, list.h, list.cpp, and listOutput.txt files.