Project 3

CSc 136

Kutztown

**Topic:** Class Design and Implementation, Updating, and Testing  
**Points:** 10 (Design Artifacts) + 5 (Updates.txt) + 20 (Code)  
**Due:** As noted below. No late submissions accepted.  
**Overview:** For this project, you will update the Array class to make it partially-filled. It will use a template function to sort itself. You will simultaneously complete an application to test the updated Array object-type.

Array Class:
The Array class, provided in the Project3 directory on acad and the web, is to be updated to partially filled. You must perform the following updates:

- Change the name of data member size to capacity. All member functions that accessed or were named using the term size now refer to capacity, and any calls of the function(s) must change as well.
- Add a data member eltsInUse that holds the number of elements in use in the object.
- Add a new function sort() that properly calls selSort() in SortSearch.h (also provided), passing it the C++ array data member and the number of elements in use.
- Add a new operator += in the class that places an int in the next available slot in Array’s C++ array member ptr. If there is no available slot, it expands the Array’s C++ array member by one element (update everything that needs it) and places the new value in the space added at the end. As an assignment operator, it returns the object.

/export/home/public/spiegel/cis136/Projects/Project3 on acad contains all files prepared for this project. They are also on the web. Copy them to your project subdirectory (make a new subdirectory for this project).

Other changes required are:

- Add necessary functions (sets/gets?), and alter functions affected. Examples of additions/alterations (this is definitely not an exhaustive list):
  - getElementsInUse(int): Get number of elements in use.
  - Copy constructor: Set #elts in use to that of the passed in object (its parameter) and have it copy over only those elements in use.
  - Stream insertion: Only output the elements in use.

Drawings

This project requires you to complete several drawings depicting how the following updated Array operations are carried out (format is up to you; the drawings must demonstrate that you understand what is required of the operation):

- Copy constructor
- Append: (operator +=) You must supply at least two drawings; one showing application of += on an Array of capacity 5 with 3 values and one for the same Array when full. Provide a before and after look, to clearly demonstrate what occurs.

Submit these by the designated deadline. Other drawings are recommended and some can gain bonus points (see below).

Class Update

Go through every function in the Array class (it doesn’t have to be in order; better you do this in the order you test) and make all necessary changes. Document each and every change you make (including those noted above) in a file named Updates.xlsx (also provided, partially completed; make the changes in there first, then in the program) as described in the notes below. This work counts for 5 of the points possible in this assignment. You must create a complete, readable document to get full credit.
Project 3

Test Driver
You are to write an application named testArray.cpp to test the updated Array class. You must provide well-labeled tests of all updated functions to demonstrate that your class update works. You may update existing examples to create this file.

Notes:
- Updates.xlsx will be used to record your additions and updates to the Array class. Either ftp it from acad or download it from the Project page on the course website.
  - Once you determine each of the updates necessary for Array to permit partial use, make the change and record it in the workbook, which will be submitted twice, as follows:
    - The initial version will be due on the project due date. You will submit it with your updated Array class for grading.
    - Updates.xlsx will be graded. You will make the revisions suggested in the grade report to it and your Array class and resubmit the program files and workbook for a final grade.
- The >> operator has been eliminated in this project.
- The [] operators will only allow access of in-use elements. To append a value to an Array, you must use the += operator that you will write.
- The Array object is to be sorted after any addition using +=.
- A makefile, the Array class example (not updated) and SortSearch.h, are available in the project’s directory on acad. There is also a complete makefile, and Updates.xlsx with the first few entries done for you.
  - You are required to work incrementally, testing updates to the Array class as you make them in testArray.cpp.
- There are no friend functions in this project. In particular, no stream operators are to be friends.
- You must upgrade all member functions’ header blocks to include member type. All functions in all files must have header blocks that include parameter types (e.g. import). Two letter grade minimum penalty for omission.
- All notes and requirements from previous projects not superseded by requirements of this project remain in force.
- All concerns or errors from previous projects noted on your grade report are to be resolved for this project. In particular, issues regarding OO design or modularity will be heavily penalized if not corrected in this project.

Deliverables:
- Drawings of copy constructor, += operator, in the appropriately named Desire2Learn dropbox. Due: 12 Noon Saturday, March 25, 2017; 10 point bonus for also including the following drawings:
  - depicting application of the assignment operator on a non-full Array object to another Array object of different size.
  - demonstrating all possible results of comparing two arrays using ==
  To get the bonus, all drawings must be complete and correct.
10 point penalty on 2nd part of project if drawings not submitted, in addition to losing the 10 points possible.
- Updates.txt in the appropriately named Desire2Learn dropbox. Due: 10 AM on Wednesday, March 29, 2017.
- Final Project: Due: 12 Noon on Friday, March 31, via turnin. Submit all .cpp files and .h files, and the makefile. There are 5 files: the updated Array class (2 files), testArray.cpp, SortSearch.h, and the makefile. Do not turn in .o files or the executable. You may also turn in a README if you want to explain anything about the overall project. The files must be named as specified previously. 2 point penalty for non-compliance. Any submission not accompanied by an Updates.xlsx file with significant work will receive a grade of 0 for the entire project.