Project 3 **CSc 136** **Kutztown**

Topic: Class Design and Implementation, Updating, and Testing
Points: 10 (Design Artifacts - Drawings) + 15 (Updates.xlsx) + 25 (Code). Bonus available.
Due: As noted in Deliverables section. Submit late at your own risk only in effect for final project.
Overview: For this project, you will update the Array class to permit it to be partially-filled. You will simultaneously complete an application to test the updated Array object-type.

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: Effect on an Array that isn't full.
- 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, both pictorially and also to the state of the object.

Read through the rest of these specs and then your first task is to complete the drawings. Submit these by the designated deadline. Drawings for each member function/operator update are strongly recommended and some can gain bonus points (see below).

Array Class:

The *Array* class, provided in the Project3 directory on acad and the web, is to be updated from fully used 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 set & get 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 tears down and rebuilds the **Array's C++ *array* member,** adding one element (update data member(s) that require 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). Names on files on the web must end with a type the server can handle; you may need to remove those suffixes, e.g. *makefile.txt* is *makefile* on acad.

Among other changes that will need to be made are adding necessary functions (sets/gets?), and alter functions affected. Examples of additions/alterations (this is definitely **not** an exhaustive list):

- getEltsInUse(): Returns the number of elements in use.
- Copy constructor: Set the elements in use to that of the passed in object (its parameter) and copy over only those elements in use from the source object.
- Stream insertion: Only output the elements in use.

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 15 of the points possible in this assignment. You must create a complete, readable document to get full credit.

Project 3 **CSc 136** **Kutztown**

Test Program

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.

A very simple application *tst.cpp* is provided to get you started. The command *make tst* will build an executable **tst** that you run using **./tst** .

Once you have enough updated so it runs copy *tst.cpp* to *testArray.cpp*. Work through the changes needed in the member functions of the **Array** class that *testArray.cpp* will be using and record them in the **Updates** workbook, and then implement them in the class.

Notes:

- Label all drawings completely. Their files must have descriptive names. Submissions whose files aren't named descriptively will not be graded.
- **Updates.xlsx** will be used to record your additions and updates to the `Array` class. Download it from the Project page on the course website. It is partially complete.
  - Once you determine each update necessary for **Array** to permit partial use, make the change and record it in the workbook. Then, implement it in the class and write code in *testArray.cpp* to test it.
- The >> operator has been eliminated from the **Array** class in this project.
- The [] operators are to allow **only** access of in-use elements. To expand an Array, you must use the += operator that you will write to append a value. Only += can increment the **eltsInUse** data member
- The *Array* class example (not updated), and tst.cpp are available in the project's directory on acad and the web. There is also a complete makefile, and Updates.xlsx with the first few entries done for you is on the web.
  - You **must** work incrementally, testing updates to the Array class in *testArray.cpp* as you make them.
  - First get the functions called by tst.cpp updated in the **Array** class. Then, to create an executable from tst.cpp, type **make tst** on the command line. The executable is run using command `./tst`.
  - The makefile creates an executable named *testArray* when you enter **make** alone on the command line.
- There are no friend functions in this project. In particular, the stream insertion operator is not to be a friend.
- 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 **Project 3 Drawings** dropbox. Deadline Sat. Oct 26 @ 1 PM. Up to a 10 point bonus for also including these drawings:
  - depicting application of the assignment operator on a non-full Array object to another Array object of different capacity.
  - demonstrating by separate examples all possible results of comparing two arrays using ==
- **No late drawings accepted. Drop box will be unavailable after deadline. You must get at least 6/10 on required drawings to be eligible to complete the remaining portions of the project.**

The final project will not be graded if your drawings are not submitted or do not meet the minimum grade threshold of 6/10.

- **Updates.xlsx** in the **Project 3 Updates** dropbox. Due with project.
- **Project Files**: Deadline TBA in class; via **turnin**. Submit .cpp files and .h files, also submit 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; **do NOT submit tst.cpp**. You may also turnin 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.