CS 260
Programming Lab 1 Part A

For this exercise you are to create a document containing pseudocode algorithms for the methods in the real lab.

Those methods are listed below for reference purposes. If you are planning on completing the entire, advanced portion of the lab, you should submit those two methods as part of this assignment.

Base Lab Methods

- Constructor (Python init) – create an array of ***n*** integers where ***n*** is the parameter passed in.If no value is passed in, create an array of 10 items (default constructor). If ***n*** is less than 1, create an array of size 10. ***Do not initialize the elements of the array*****.** (Python requires initializing the array, use zero).
- int getSize() – return the current size of the array
- void resize(int size) – resize the array to the new **size**, copying values as needed. If the newsize is less than or equal to the current size, do nothing. You are not to use built in resize methods in C# or Python.
- void append(int value) – add ***value*** at the next available location in the array. If the next location is greater than or equal to the current size of the array, create a new array of size*2 and copy all values to it. The first value should be added at index 0 and so forth. This will require keeping track of the next available location.
- int getLast() – return the value at the previously available location. If the array is empty, throw an ***out_of_range*** exception (Python use ***IndexError***).
- void deleteLast() – similar to getLast, but deletes the last item and does not return anything. Note that deleting an item means making it not available. This does not require setting it to a default value or doing a C++ delete. If the array is empty, do nothing.
- string listElements() – return a string containing a comma-separated list of the elements inthe array from location zero through the largest location containing valid data. If the arrayis empty, return the string “Empty Array”.
- void insertAt(int index, int value) – insert ***value*** at ***index*** location, shifting all values at that location or higher up one. If this shift would cause elements to be stored past the end of the array, create a new array of size * 2 and copy all values to it. If index is less than 0, greater than the largest location containing valid data, or greater than the array size, throw an ***out_of_range*** (***IndexError***) exception.
- int removeAt(int index) – save the value at index location, shifting all values at that location or higher down one position. Return the saved value. If index is less than 0, greater than the largest location containing valid data, or greater than the array size, throw an ***out_of_range***(***IndexError***) exception

- bool find(int value) – return true if value is in the array, false otherwise.
- bool findRemove(int value) – if value is in the array, remove it while maintaining the samearray order. Return true if value is in the array, false otherwise.
- int findLargest() – return the largest element in the array. If the array is empty, throw an ***out_of_range*** exception (***IndexError***).
- void removeLargest() – remove the largest element in the array, while maintaining the same array order. If the array is empty, throw an ***out_of_range*** exception (***IndexError***)

Advanced Lab Methods

- int getAt(int index) – return the value at **index** location. If **index** is less than 0 or greaterthan the array size, throw an ***out_of_range*** exception (***IndexError***). This may return a value from a location that was not previously set.
- void setAt(int index, int value) – store **value** at **index** location. If **index** is less than 0 or greater than the array **size**, throw an ***out_of_range*** exception (***IndexError***). Make sure that any following appends will not overwrite this value. This might change a value that was previously set or it might store the value at a location at an index higher than any previous append or insert has used.