COSC 251 – Programming Languages
Project 1
Spring 2014

Objective: Implement a data structure in C/C++ that is easily dealt with in JAVA.

Your Task: You will create an Object Oriented version of a dynamic array data structure that we will review in class (similar to ArrayList in Java). A typical array in C++ is of fixed, non-variable length. For the data structure you create, you will allow for dynamic extension of the array. To facilitate this, you will overload certain operators that are used for normal arrays. Some notes:

- Attempting to access an array cell that is currently out of bounds should resize the array, then return the value in that cell. Note that this will be a rubbish value.
- As implied in the above note, there should be no default values for any cells in the array.
- Assigning a value to a cell that is currently out of bounds should resize the array, then return the value in that cell.
- Upon deletion (see delCell), you should resize the array down to the currently used size, down to a minimum of the initial size of the array. How you handle determining what cells are being used or not is up to you.
- For this structure, we assume that we will only be storing ints.

You may use any internal structure for your array storage that you wish.

For this project you will be provided a separate driver file that will run a variety of test cases on your code. To interface with the driver file you must match the following signatures/names exactly:

Class – DynArray
Constructor – DynArray(int i) //initial size of the array
Constructor – DynArray() //initial size of 0
Constructor – DynArray(const DynArray &) //copy constructor
Function – int size() //returns current size of the array
Function – void delCell(int i) //deletes the cell at index i

You are also required to overload the following operators:

= (assignment)
[ ] (array notation, needs two – one for left hand side of an expression, one for right hand side)
<< (allows for printing using cout)
>> (allows for input using cin)

+= (adds the element on the rhs to the lhs dynamic array)

Deliverables: the source files for your dynamic array. If your code does not work with the provided driver, you will earn a 0 for this project.

Learning Targets: What I hope that you get out of this project is a firm understanding of the basics of C++ programming. This is also an exercise in classic data structure based problem solving as I am leaving many of the implementation details up to you. Finally, this is an information literacy exercise as we will not be going over operator overloading in class (there are many online resources on this topic).

Expectations: The code should be clean, concise, well-commented and correct. If you use an outside source, be sure to document that source. Significant use of outside sources will result in a deduction. Grading rubric will be provided a week ahead of the due date. You may work in pairs on this project. If you are working in a pair, one member of the pair is required to email me names by 5pm on February 5th, no exceptions.

DUE: February 19th, 11:59pm via Blackboard