Lab #3 – Copy Constructors, Assignment Operators, Destructors

Handed out: Tue, 2/23/2016	Due date: Tue, 3/1/2016

Goal

The goal of this assignment is to obtain initial understanding of copy constructors, assignment operators, and destructors.

Submission instructions

Inside that the homework subdirectory, create a directory for homework #3, and call it `lab3`. When you finish the assignment, go to the homework directory and submit it as follows:

```
$ submit jwang lab3 lab3
```

Problem

Your assignment is to implement a dynamic array class `MyDynArray`. The class should provide the functionality defined in `MyDynArray.h`.

You should also supply the testing code that fully tests the functionality that you have implemented. The testing code should be placed in the file `MyDynArray_user.cc`.

You should submit three files: the original MyDynArray.h, your implementation in `MyDynArray.cc` and your testing code in `MyDynArray_user.cc.`

Your implementation should follow the specification in `MyDynArray.h` (see page 2).

Note that the function `MyDynArray::set(T, size_t)` should change the size of the array in case the position requested falls outside of the current array size. If the request succeeds, the function should return `true`; if the request fails, the function should return `false`.

Sample initial test code is available in `MyDynArray_user_sample.cc.` Feel free to use it as the starting point, or to write your own.

For testing purposes, your implementation of each member function should report when it is invoked, so when you run your test code the output should look something like what is shown below (see page 3).

```
#ifndef MYDYNARRAY_H
#define MYDYNARRAY_H

typedef int T;  // specify the data type to be stored in the array

class MyDynArray {

 private:

  // number of elements currently in the array
  size_t size;

  // array pointer
  T *array_ptr;

 public:

  // constructor; default size is 100 items
  MyDynArray(size_t size_ = 100);

  // get the current array size
  size_t getSize() const;

  // put the element at the position specified by index
  // if the position is out of range, increase the size of array accordingly
  bool set(T element, size_t index);

  // get the value at the position specified by index
  T get(size_t index) const;

  // copy constructor: should do a deep copy
  MyDynArray(const MyDynArray& arg);

  // assignment operator: should do a deep copy
  MyDynArray& operator=(const MyDynArray& rhs);

  // destructor
  ~MyDynArray();
};
#endif
```

```
***** Testing copy constructor *****
Constructor called with: 10
Copy constructor called
a: 0 0 0 0 0 0 0 0 0 0
Destructor called
Function set() called with element: 3 and index: 9
Copy constructor called
Copy constructor called
b: 0 0 0 0 0 0 0 0 0 3
Destructor called
Function set() called with element: -2 and index: 9
Copy constructor called
b: 0 0 0 0 0 0 0 0 0 -2
Destructor called
Copy constructor called
a: 0 0 0 0 0 0 0 0 0 3
Destructor called
PASSED: Copy constructor
***** Testing assignment operator *****
Constructor called with: 100
Assignment operator called
Copy constructor called
c: 0 0 0 0 0 0 0 0 0 3
Destructor called
Function set() called with element: -4 and index: 3
Copy constructor called
c: 0 0 0 -4 0 0 0 0 0 3
Destructor called
Copy constructor called
a: 0 0 0 0 0 0 0 0 0 3
Destructor called
PASSED: Assignment operator
Destructor called
Destructor called
Destructor called
```

Grading

The assignment will be graded as follows:

- Constructor: 2 pts
- Destructor: 2 pts
- Assignment: 4 pts
- Copy constructor: 4 pts
- Set: 4 pts
- Testing: 3 pts
- Style, comments, other: 1