ET-580 - Pointers & Classes - Homework

Reading

1) Chapter 14 Inheritance

Implementation

1. Implement a partially filled array class *IntArr* using dynamic memory.

   a. data members:
   capacity: maximum number of elements in the array
   size: current number of elements in the array
   array: a pointer to a dynamic array of integers

   b. constructors:
   default constructor:
      1. Set *capacity* to 5, *size* to 0
      2. Create a dynamic array of *capacity* 5

   user constructor:
      1. Parameter *c* to set the *capacity*
      2. Set *capacity* to *c*, *size* to 0
      3. Create a dynamic array of *capacity c*

   c. overloaded operators:
   subscript operator:
      1. Parameter *index*
      2. Return the element at position *index*

   constant subscript operator:
      1. Identical to subscript operator but for constants
      2. This is required for the big three to work properly

   d. the big three - copy constructor:
      1. Construct a new *IntArr* object from an existing *IntArr* object

   e. the big three - assignment overload operator:
      1. Copy from an *IntArr* object to another *IntArr* object

   f. the big three - destructor:
      1. Destroy the dynamic memory of and IntArr object

   g. grow function:
      1. This should be a private function (located in private).
      2. Grow the array so that the new capacity is original capacity*2+1

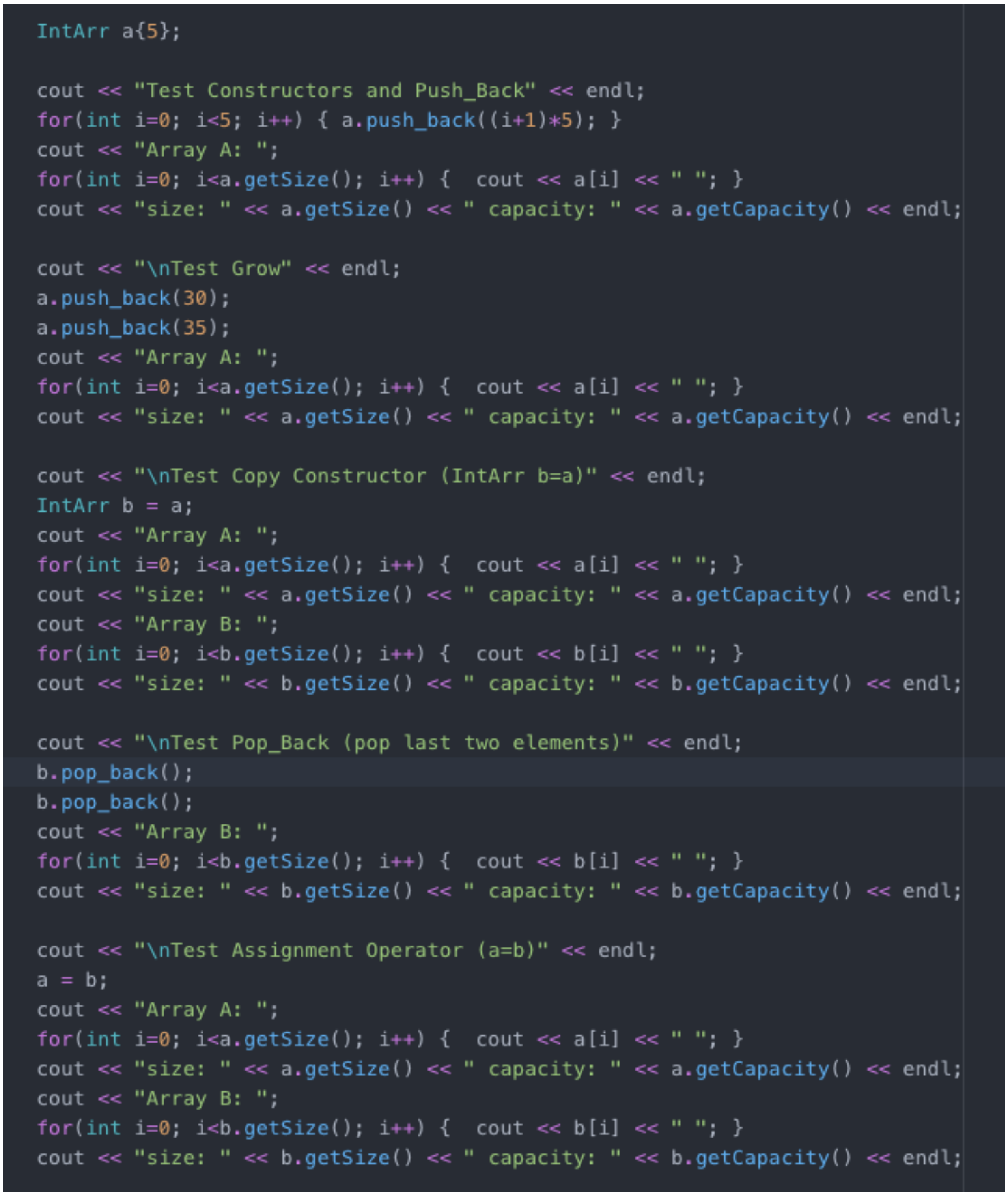
   h. push_back function:
      1. If size is equivalent to capacity call the grow function
      2. Append a new integer to the end of the array

   i. pop_back function:
      1. If the array is not empty, decrement size by one

   j. getSize function:
      1. return the current size of the array

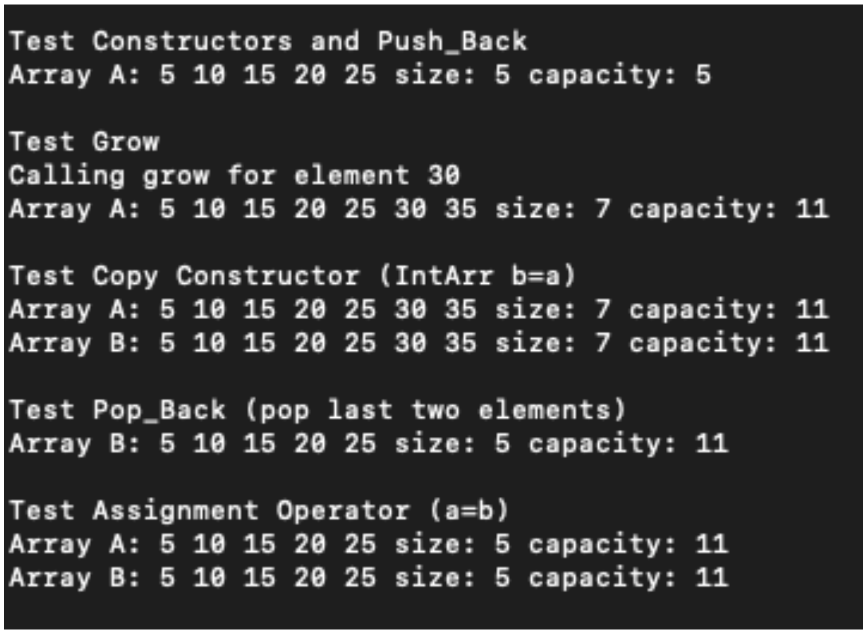
   k. getCapacity function:
      1. return the current capacity of the array

Main Function



```
IntArr a{5};

cout << "Test Constructors and Push_Back" << endl;
for(int i=0; i<5; i++) { a.push_back((i+1)*5); }
cout << "Array A: ";
for(int i=0; i<a.getSize(); i++) {  cout << a[i] << " "; }
cout << "size: " << a.getSize() << " capacity: " << a.getCapacity() << endl;

cout << "\nTest Grow" << endl;
a.push_back(30);
a.push_back(35);
cout << "Array A: ";
for(int i=0; i<a.getSize(); i++) {  cout << a[i] << " "; }
cout << "size: " << a.getSize() << " capacity: " << a.getCapacity() << endl;

cout << "\nTest Copy Constructor (IntArr b=a)" << endl;
IntArr b = a;
cout << "Array A: ";
for(int i=0; i<a.getSize(); i++) {  cout << a[i] << " "; }
cout << "size: " << a.getSize() << " capacity: " << a.getCapacity() << endl;
cout << "Array B: ";
for(int i=0; i<b.getSize(); i++) {  cout << b[i] << " "; }
cout << "size: " << b.getSize() << " capacity: " << b.getCapacity() << endl;

cout << "\nTest Pop_Back (pop last two elements)" << endl;
b.pop_back();
b.pop_back();
cout << "Array B: ";
for(int i=0; i<b.getSize(); i++) {  cout << b[i] << " "; }
cout << "size: " << b.getSize() << " capacity: " << b.getCapacity() << endl;

cout << "\nTest Assignment Operator (a=b)" << endl;
a = b;
cout << "Array A: ";
for(int i=0; i<a.getSize(); i++) {  cout << a[i] << " "; }
cout << "size: " << a.getSize() << " capacity: " << a.getCapacity() << endl;
cout << "Array B: ";
for(int i=0; i<b.getSize(); i++) {  cout << b[i] << " "; }
cout << "size: " << b.getSize() << " capacity: " << b.getCapacity() << endl;
```

Output



```
Test Constructors and Push_Back
Array A: 5 10 15 20 25 size: 5 capacity: 5

Test Grow
Calling grow for element 30
Array A: 5 10 15 20 25 30 35 size: 7 capacity: 11

Test Copy Constructor (IntArr b=a)
Array A: 5 10 15 20 25 30 35 size: 7 capacity: 11
Array B: 5 10 15 20 25 30 35 size: 7 capacity: 11

Test Pop_Back (pop last two elements)
Array B: 5 10 15 20 25 size: 5 capacity: 11

Test Assignment Operator (a=b)
Array A: 5 10 15 20 25 size: 5 capacity: 11
Array B: 5 10 15 20 25 size: 5 capacity: 11
```