Homework 4: Dynamic Arrays

Due: 11:59 PM, March 5, 2021

In this assignment, you will implement and use Dynamic Arrays – the focus of our past couple of lectures. In the first exercise you will implement your own `DynamicArray` class and you will then use it in the second exercise. In the third exercise you will use the C++ STL `vector` class (which our `DynamicArray` class is modeled after).

Exercise 1 (70 points): Implementing a DynamicArray

Your task is to implement the following methods within the `DynamicArray` class.

```
template <typename T>
class DynamicArray {
public:
  DynamicArray() { /* TODO: Implement me */ }

  DynamicArray(int initialSize) { /* TODO: Implement me */ }

  DynamicArray(int initialSize, T defaultValue) { /* TODO: Implement me */ }

  DynamicArray(const DynamicArray<T> &da) { /* Code Provided */ }

  ~DynamicArray() { /* Code Provided */ }

  DynamicArray<T>& operator= (const DynamicArray<T> &da) { /* Code Provided */ }

  T& operator[] (int index) { /* Code Provided */ }

  void push_back(T item) { /* TODO: Implement me */ }

  void pop_back() { /* TODO: Implement me */ }

  int size() { /* TODO: Implement me */ }

  int capacity() { /* TODO: Implement me */ }

  void clear() { /* TODO: Implement me */ }

  void reverse() { /* TODO: Implement me */ }
};
```

Descriptions of each of the public methods are listed below:

- `DynamicArray()`: This constructor initializes the `DynamicArray` with a size of zero and a capacity of 8.
- `DynamicArray(int initialSize)`: This constructor initializes the `DynamicArray` with a size of the provided `initialSize` and a capacity that is double the initial size. Don't worry about setting the values of the first `initialSize` items – allow them to retain their "garbage" values but do consider them as items within the `DynamicArray`.
- `DynamicArray(int initialSize, T defaultValue)`: This constructor initializes the `DynamicArray` with a size of the provided `initialSize` and a capacity that is double the initial size. Assign each of the first `initialSize` items a value equal to the provided `defaultValue`.
- `DynamicArray(const DynamicArray<T> &da)`: This constructor is implemented for you and it is known as the "copy constructor." It will make one `DynamicArray` from another. The code I provide for you copies each item in the provided array over to the new `DynamicArray` we initialize with this constructor. It is important that this new `DynamicArray` creates its own array that lives in the heap that we copy the data to because each should have their own space. We don't want to just make this new `DynamicArray` point to the other's data.
- ~`DynamicArray()`: Deallocates any heap-allocated space belonging to the `DynamicArray`.
- `void push_back(T item)`: Adds the value in `item` to the end of the `DynamicArray` increasing size and capacity as necessary.
- `void pop_back(T item)`: Removes the last item in the `DynamicArray`.
- `int size()`: Returns the current size of the `DynamicArray`.
- `int capacity()`: Returns the current capacity of the `DynamicArray`.
- `T& operator[] (int index)`: Returns a reference to the item at position `index` so that it can be accessed and modified through the familiar square-bracket notation. Don't worry about trying to protect from people accessing items that don't exist (e.g., `myArray[-3]` or `myArray[9999999]`). Note: I have completed this function for you which you can see in the referenced header file on CS-Data.
- `DynamicArray<T>& operator= (const DynamicArray<T> &da)`: This is the assignment operator and it allows you to assign the contents of one `DynamicArray` to another `DynamicArray` replacing its original contesnts. In the code I provide to you, we delete the old array, make a new array of the size that we are receiving, and copies all of the values from the target array to the destination array. It is important that we allocate new space for the target array so that the two `DynamicArray`s have their own data and don't point to the same place in the heap. Each should have their own space.
- `void clear()`: Removes all items from the `DynamicArray` resulting in a `size` of zero.
- `void reverse()`: Reverses the order of all items within the `DynamicArray` – the last item becomes first and the first item becomes last.

I have provided you all with a default header file for the `DynamicArray` class. It includes many comments to help guide you in implementing the methods. You can find it at:

\\CS-Data\Courses\cmsc182\homework\hw_4\DynamicArray.h

Please start with this file and implement the methods within it. It is vital that you use exactly this file and implement the methods exactly as they have been defined or I will not be able to use the `DynamicArray` that you have written.

With this file completed, you should also test the functionality of your `DynamicArray` within your own `ex_1.cpp` file that you create. Ensure that each of your methods works appropriately by testing them from within this `ex_1.cpp` file.

With these two files (you'll find a starter `ex_1.cpp` on CS-Data as well), I have given you working but not-fully-implemented code. That is, you can compile it now – it just lacks in functionality. Compile and test often to ensure that you don't get in over your head with compilation errors.

Exercise 3 (20 points): Sieve of Eratosthenes - DynamicArray

Use your `DynamicArray` to implement a Sieve of Eratosthenes that identifies and then prints to the screen all prime numbers less than 100.

To acquaint yourself with the Sieve of Eratosthenes, please see the Wikipedia article linked below and view the *Example* section as well as the *Pseudocode* section.

http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

Exercise 3 (10 points): Sieve of Eratosthenes - Vector

Use the `vector` class provided by the C++ STL library to implement a Sieve of Eratosthenes that identifies and then prints to the screen all prime numbers less than 100.

Hint: this *should* just required that you change the `#include` to reference `<vector>` instead of "`DynamicArray.h`" and changing all of your references of `DynamicArray` to `vector`.

Submission: Turning in the assignment

After completing all of the exercises in this homework assignment, you should have the following files/folders saved in your CS-Data folder:

\\CS-Data\Students\your_user_name\cmsc182\hw_4\DynamicArray.h
\\CS-Data\Students\your_user_name\cmsc182\hw_4\ex_1.cpp
\\CS-Data\Students\your_user_name\cmsc182\hw_4\ex_2.cpp
\\CS-Data\Students\your_user_name\cmsc182\hw_4\ex_3.cpp
This is what I will be looking for to grade. If you do not have these files/folders, something went wrong – come and contact me.