

Homework – Stacks

Brad Peterson – Weber State University

The goals of this assignment are to:

- Help review C++.
- To better understand classes, arrays, dynamic allocation and deallocation, error handling, templates, and basic logic.
- To help you prepare for the style of future homework assignments.

For this assignment **complete the class called `StackForCS2420`. It should be a template class**, meaning you would prefix the class and all class members declared outside the class with `template <typename T>`. (Note, the book uses `template <class T>`, which is the exact same thing. The keyword `typename` is newer and better.)

For my unit tests, I made a base class with methods that simply have enough logic to compile. **You should not modify the base class**. Instead, **you should modify the derived class and override the constructor, destructor, and all methods there** (I gave you an example of overriding the constructor's declaration).

Note that you should declare `arr` in the derived class instead of the base class defined in the base class. (If you declare it in the base class, you should always use `this->arr` instead of just `arr` to make compilers happy.)

The `StackForCS2420` class needs to have the following members:

- A **private unsigned int `index` data member**, which keeps track of the next open index in the array.
- A **private unsigned int `capacity` data member**.
- A **constructor that accepts a const unsigned int parameter**. This constructor needs to dynamically allocate an array of the size passed into the parameter. Use the `new` keyword to make this array. The constructor should set the data member `index` to zero (only if you haven't default initialized it to zero). It should also set the `capacity` data member to the value of the argument passed in.
- You need a destructor because you used the `new` keyword in the constructor.
- A **`size()` method**. The return type is unsigned int. It returns the value of `index`.
- A **`push()` method**. This method should have a single parameter, the data type of that parameter should be `const T&`. The `const` means it can't be changed. The `&` means it will be passed in by reference (instead of by value, which makes a copy). **The `push()` method should have a void return value. This method should see if the `index` equals the `capacity` (seeing if it is full). If so, simply state an error message and return. Otherwise, insert the value into the array at the correct spot, and increment `index`.**
- A **`pop()` method**. This method should not have any parameter. The return type should be void. The purpose of the method is to "pop" the item off the stack. It doesn't actually pop the item off the array, it just changes `index`.
- A **`top()` method**. This method should not have any parameter. The return type should be `T`. It should return what is at the top of the stack. It should first check if `size` is zero. If so, then the stack is empty, so throw an error with `throw std::out_of_range("some message");`. Otherwise, return the correct value.
- A **`popThirdFromTop()` method**. As the name implies, pops the item underneath the top two items. (Note, stacks typically aren't used this way, but this method is here to help you extend your understanding of both stacks and programming.)
- A **`pushTwoUnderTop()` method**. As the name implies, pushes an item under the top two items.
- A **`topThirdFromTop()` method**. As the name implies, retrieves the value of the item under the top two items. Throws an error with `throw std::out_of_range("some message");` if this isn't possible.

Use the `.cpp` file given. Your assignment should pass all tests.