Homework – Stacks

Goals:

- 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.

Overview:

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).

Completing the Class:

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. Initialize it to zero.
- A private unsigned int `capacity` data member, initialized to zero.
- A private `T* arr` data member, initialized to `nullptr`.
- A constructor that accepts a const unsigned int parameter for the capacity. The constructor should also set the `capacity` data member to the value of the argument passed in. This constructor also needs to dynamically allocate an array of the size passed into the parameter. Use the new keyword to make this array.
- A destructor which reclaims the memory allocated 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 (`throw 1;`). Otherwise, return the correct value.
- A `popThirdFromTop()` method. As the name implies, pops the item underneath the top two items. (Note, stacks typically are not 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 (`throw 1;`) if this is not possible.

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

Sample Output:

```
Passed testStackForCS2420 #1
Passed testStackForCS2420 #2
Passed testStackForCS2420 #3
Passed testStackForCS2420 #4
Passed testStackForCS2420 #5
Passed testStackForCS2420 #6
Passed testStackForCS2420 #7
Passed testStackForCS2420 #8
Passed testStackForCS2420 #9
Passed testStackForCS2420 #10
Passed testStackForCS2420 #11
Passed testStackForCS2420 #12
Passed testStackForCS2420 #13
Passed testStackForCS2420 #14
Passed testStackForCS2420 #15
Passed testStackForCS2420 #16
Passed testStackForCS2420 #17
Passed testStackForCS2420 #18
Passed testStackForCS2420 #19
Passed testStackForCS2420 #20
Passed testStackForCS2420 #21
Passed testStackForCS2420 #22
Passed testStackForCS2420 #23
Passed testStackForCS2420 #24
Passed testStackForCS2420 #25
Press enter to continue...
Passed testStackAdditional #1
Passed testStackAdditional #2
Passed testStackAdditional #3
Passed testStackAdditional #4
Passed testStackAdditional #5
Passed testStackAdditional #6
Passed testStackAdditional #7
Passed testStackAdditional #8
Passed testStackAdditional #9
Passed testStackAdditional #10
Passed testStackAdditional #11
Passed testStackAdditional #12
Passed testStackAdditional #13
Passed testStackAdditional #14
Passed testStackAdditional #15
Press enter to continue...
```

Submission:

Just submit your .cc or .cpp file.