Exercise - Dynamic Array in C++

The idea of this exercise is to create a dynamic array that stores integers (`int`) in C++.

The following is the expected output for the code below.

```
Size: 1 Capacity: 1 Contents: [10]
Size: 2 Capacity: 2 Contents: [10, 20]
Size: 3 Capacity: 4 Contents: [10, 20, 30]
Size: 4 Capacity: 4 Contents: [10, 20, 30, 40]
Size: 5 Capacity: 8 Contents: [10, 20, 30, 40, 50]
Element at index 1: 20
Modified element at index 1: 25
After insertion at index 1: 15
After deletion at index 2: 30
After pop_back(): Size is now 4
Size: 4 Capacity: 8 Contents: [10, 15, 30, 40]
```

C++ Dynamic Array Template

Copy the following into **main.py** and fix the following methods so that they perform as expected:

- `insert()`
- `erase()`
- `push_back()`
- `pop_back()`

A correct implementation will result in the output above. Note that you don't need to worry about exceptions etc, but by all means add them in.

I have implemented a helper method called `resize()` that you should use in your `insert()` method to resize the array when it is full.

Additional Task

Add another member variable to the class that stores the threshold at which to shrink the array. e.g. `1/3` would see you shrink an array of size `30` when you reach `10` items (.i.e. a third of `3`). This would then shrink the array to size `15` as the growth factor is `2`, so shrink by `1\2` (a half).

I would love to see some implementations, so please show me if you manage to do this 😄

```
#include <iostream>
#include <stdexcept>
#include <sstream>

class DynamicIntArray {
```

```cpp
private:
    int* arr;            // Pointer to the dynamic array
    int size;            // Current number of elements in the array
    int capacity;        // Current capacity of the array
    int initialCapacity = 1;
    double GrowthFactor = 2.0;

public:
    // Constructor with default capacity and growth factor
    DynamicIntArray() {
        // Allocate a new array with the specified initial capacity
        arr = new int[initialCapacity];

        // Set the initial size to 0, as the array is empty
        size = 0;

        // Set the initial capacity to the specified value
        capacity = initialCapacity;
    }

    // Destructor to free allocated memory
    ~DynamicIntArray() {
        delete[] arr;
    }

    // Get the element at the specified index
    int get(int index) const {
        // Check if the index is within bounds
        if (index < 0 || index >= size) {
            throw std::out_of_range("Index out of bounds");
        }
        return arr[index];
    }

    int getCapacity(){
        return capacity;
    }

    int getSize(){
        return size;
    }

    // Set the element at the specified index to the given item
    void set(int index, int item) {
        // Check if the index is within bounds
        if (index < 0 || index >= size) {
            throw std::out_of_range("Index out of bounds");
        }
        arr[index] = item;
    }

    // Insert the given item at the specified index
    void insert(int index, int item) {
    }

    // Delete the element at the specified index
    void erase(int index) {

    }

    // Convert the contents of the array to a string
    std::string toString() const {
        std::stringstream ss;
        ss << "[";
        for (int i = 0; i < size; ++i) {
            ss << arr[i];
            if (i < size - 1) {
                ss << ", ";
            }
        }
        ss << "]";
        return ss.str();
    }

    // Add an item to the end of the array
    void push_back(int item) {
        // check if at capacity and resize if required
        // add the item to the end of the array, you should consider which
variable tracks the end of the current items in the array
        // make sure you increment this variable by 1
    }

    // Remove the last item from the array
    void pop_back() {
        // Check if the array is not empty and decrement the size to effectively
"remove" the last item

    }

private:
    // Helper function to resize the array
    void resize() {
        // Calculate the new capacity based on the growth factor
        int newCapacity = static_cast<int>(capacity * GrowthFactor);

        // Allocate a new array with the updated capacity
        int* newArr = new int[newCapacity];

        // Copy elements from the old array to the new array
        for (int i = 0; i < size; ++i) {
            newArr[i] = arr[i];
        }

        // Free the memory occupied by the old array
        delete[] arr;

        // Update the pointer and capacity
        arr = newArr;
        capacity = newCapacity;
    }
};

int main() {
    // Example usage
    DynamicIntArray myArray;

    // Adding elements to the array
    myArray.push_back(10);
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;
    myArray.push_back(20);
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;
    myArray.push_back(30);
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;
    myArray.push_back(40);
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;
    myArray.push_back(50);
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;

    // Accessing elements
    std::cout << "Element at index 1: " << myArray.get(1) << std::endl;

    // Modifying elements
    myArray.set(1, 25);
    std::cout << "Modified element at index 1: " << myArray.get(1) << std::endl;

    // Inserting an element
    myArray.insert(1, 15);
    std::cout << "After insertion at index 1: " << myArray.get(1) << std::endl;

    // Deleting an element
    myArray.erase(2);
    std::cout << "After deletion at index 2: " << myArray.get(2) << std::endl;

    // Removing the last element
    myArray.pop_back();
    std::cout << "After pop_back(): Size is now " << myArray.getSize() <<
std::endl;
    std::cout << "Size: " << myArray.getSize() << " Capacity: " <<
myArray.getCapacity() << " Contents: " << myArray.toString() << std::endl;

    return 0;
}
```