the proceeding size increase attempt. This is done in a proportional way because we always want a decrease in
increase size to be large enough that it could possibly allow the new allocation to succeed but not so large that the
increase in size is uselessly small.

2) No attempt to recover will be made by trying to resize to a different size. This is because trying a larger size
won't possibly make the problem better. Also, this is an assignment specification.

3) Since the function may not always be able to perfectly resize the array to the requested size, it must
communicate the final size somehow. However, this information is already available to the client by querying the
RobustArray::size() function. So this information doesn't need to be returned from the function call in any way.

Additional Notes
The RobustArray constructor and operator= throw bad_alloc upon memory allocation failure instead of
CouldNotAllocateExcept. This is because these two functions give up immediately instead of trying to recover
from bad_alloc.

Question 3:

Part A:
A tree store offers a variety of trees for XMas holidays. Customers can inspect the tree store's 'menu' to see the
combinations of trees and decorations that are available. The customer's can buy any combination that is available
on the menu. I intend to use the Decorator pattern for this problem. Without the Decorator pattern, each
combination of tree and decorations that the store sells and stocks would be a different type. The details of my
implementation using the Decorator pattern are shown in Part B.

Problem Without Decorator Pattern:
Note: not all possible classes are shown in the figure below. For every new combination of trees and decorations
the store wants to sell, a new class would be added to the figure below.
Part B:
You can see that all the original subclasses of Tree have been removed. In there place, there are now three concrete classes representing tree plant types and a second abstract class for decorations that can be added to the plants. There are three concrete types of decorations inheriting from this abstract decoration class.

Problem With Decorator Pattern: