Q2b, RobustArray recovery methods

at()
When FlexArray::at() throws. Robust array recovers from this exception and print the error message to std::cerr. The value returned is the minimum value of integer.
This design decision was made because the client needs to know that the accessors failed somehow. at() returns the minimum value of int type. This int is unlikely to be used correctly by the client, so it would result in an error later on as well if the client is programming defensively.

assign()
When assign() is out of range. RobustArray will resize the array and reassign it such that the assign can be done. If the resize is unsuccessful, the program will terminate. This decision is made because when the user want to assign a value out of range, he/she might just be too lazy to resize manually. And if the array resizes for the client. Development can be faster. Usually, a resize operation is possible, so exiting the program prematurely is very unlikely.

resize()
Resize attempt to to resize to a smaller size twice. If the smaller size cannot be achieved, it will through an exception with the smallest size it attempted with. This design decision was made by the assignment. It makes sense to do this because in certain circumstances it is okay to not get to requested size. If the resize still fails after several operations, a new exception is through with more detailed information on the requested size. This way, client can know if something horribly goes wrong too.

Q3 a)

Problem Description:
We have a program used by the Yum! Brand to keep track of businesses opened under the name of KFC, Taco Bell or Pizza Hut. Initially, the UML diagrams shows that there are three types of restaurants, KFC, Taco Bell and Pizza Hut. When someone opens a new restaurant, the program creates a new KFC, Taco Bell, or Pizza Hut object. If the brands under Yum change, for example, if we want to be able to create a restaurant under A&W, the client code needs to change. Applying the factory design pattern to the initial UML diagram to solve this problem.

![UML Diagram](image)

b) The UML diagram after applying the factory pattern, additional classes are highlighted.