# CS 36C Winter 2024 Homework 2:
# A Dynamically Sized Array

This project has you writing a significant implementation of a dynamically sized array. This assignment is due on January 30th at 1400 PST (6:00 pm Davis time). You must work on this assignment individually. We have provided you a rich skeleton project in **`Homework2.zip`** available from Canvas which includes a full Gradle and IntelliJ environment, a couple of non-trivial test functions, and the **`DynamicArray.ks`** and **`DynamicArrayTest.ks`** files you need to extend to implement the project.

## Expected Project Outcomes

1. Implement the major Dynamic Array functionality including appending on either end, insertion, accessing of arbitrary elements, iterations, and higher order procedures
2. Implement a comprehensive test suite for your Dynamic Array implementation

## Implementation and Grading

You should implement the specified function in DynamicArray.ks. These are:

1. **`DynamicArrayIterator.computeNext() : Unit`**
2. **`DynamicArray.append(item: T) : Unit`**
3. **`DynamicArray.prepend(item: T) : Unit`**
4. **`DynamicArray.get(index: Int) : T`**
5. **`DynamicArray.set(index: Int, item: T) : Unit`**
6. **`DynamicArray.insertAt(index: Int, item: T) : Unit`**
7. **`DynamicArray.removeAt(index: Int, item: T) : T`**
8. **`DynamicArray.indexOf(item: T): Int`**
9. **`DynamicArray.<R>fold(initial: R, operation: (R, T) -> R) : R`**
10. **`DynamicArray.<R>map(operation: (T)->R): DynamicArray<R>`**
11. **`DynamicArray.mapInPlace(operation: (T)->T): DynamicArray<T>`**
12. **`DynamicArray.filter(operation: (T) -> Boolean): DynamicArray<T>`**
13. **`DynamicArray.filterInPlace(op: (T) -> Boolean): DynamicArray<T>`**

You should also implement test functions in **`DynamicArrayTest.ks`** for all these functions. For grading on Gradescope you will submit your **`DynamicArray.ks`** file. Your **`DynamicArray.ks`** file will be combined with our own tests to grade your DynamicArray code.

This time we are not grading your test cases, because your test cases should be very similar to those in the Linked List homework 1.

In fact, it is strongly advised that you start by simply copying large portions of your Linked List test code into the DynamicArrayTest.ks file, replacing every LinkedList reference with a DynamicArray reference, and try to run the test file. Once the test file compiles, only then should you start coding in the DynamicArray.ks file itself.

This should encourage what is known as “red to green” development.  Once you have the test cases they show up in red when they don’t work, and as you code you rerun your tests until they pass.

All functions should be linear time except that **`prepend`** and **`append`** should be (amortized) constant time, and insertion and deletion from the front and back should also be constant time, resulting in **`push`**, **`pushLeft`**, **`pop`**, and **`popLeft`** all being constant time operations.  We have included a test-case demonstrating how we evaluate performance: it first does 1000 calls to append and then does 100,000, ensuring that the latter is not significantly greater than 100 times the former.