**Structured Programming Project – Design Document**

**Module: Software Development Year 1  
Semester: 2, 2025  
Team Members: Maksym Shevchuk, Dylan Ryan Gleeson, Luka Simic Kokot**

**Team Member Contributions**

***Maksym Shevchuk***

1. Implemented the following functions:

* ScaleUp: Multiplies each element in the array by a user-defined factor.
* Reverse: Reverses the order of elements in the array.
* ZeroBase: Adjusts all values so that the smallest becomes zero.
* RemoveNumber: Removes an element at a specified position.

1. Adapted the application to read up to 50 integers from Numbers.dat (Part B).

***Dylan Ryan Gleeson***

1. Implemented the following functions:

* GetAverage: Calculates the average of the array elements.
* GetNumOccurrences: Counts how many times a specific number appears in the array.
* GetSmallest: Finds the smallest number in the array.
* ShowMenu: Displays the menu options to the user.

***Luka Simic Kokot***

1. Implemented the following functions:

* Display: Shows the contents of the array.
* GetTotal: Calculates the sum of the array elements.
* GetLargest: Finds the largest number in the array.
* Sort: Sorts the array in ascending order.

**Personal Reflection – Maksym Shevchuk**

Working on this project was a valuable experience. I was responsible for implementing the following functions:​

* **ScaleUp**
* **Reverse**
* **ZeroBase**
* **RemoveNumber**
* Adaptation for Part B (reading up to 50 integers from a file)​

**Challenges and Solutions:**

* **ScaleUp and Reverse:** These functions were straightforward to implement, as we had done similar tasks in the previous semester. The main difference was that, this time, the data was read from a file.​
* **RemoveNumber:** Initially, I attempted to remove an element by finding its value and deleting it, similar to how it's done in Python using the remove() function. However, in C++, arrays have a fixed size, and there's no built-in function to remove an element directly. To solve this, I shifted all elements after the specified index one position to the left and decreased the array size by one.​
* **ZeroBase:** At first, I tried to find the minimum value manually within the function. Later, I realized that Dylan had already implemented a GetSmallest() function. However, since his function was of type void, it didn't return any value. I modified it to return the smallest value so that I could use it in my ZeroBase function.​
* **File Reading (Part B):** I adapted the program to read up to 50 integers from the "Numbers.dat" file. I implemented a counter to keep track of how many numbers were read. This way, if the file contained fewer than 50 numbers, the program would still function correctly without errors.​

**Testing Example:**

To test the RemoveNumber function, I used an array with the following elements:​

[10, 20, 30, 40, 50]​

I removed the element at position 3 (which is 30). After the operation, the array became:​

[10, 20, 40, 50]​

This confirmed that the function worked as intended.​

**Conclusion:**

This project enhanced my understanding of array manipulation, file input/output operations, and problem-solving in C++. It also highlighted the importance of adapting existing functions and collaborating effectively with team members and I created a repository on github so that during the holidays we can finish our project online.

This project enhanced my understanding of array manipulation, file input/output operations, and problem-solving in C++. It also highlighted the importance of adapting existing functions and collaborating effectively with team members and I created a repository on github so that during the holidays we can finish our project online. link to our github repository: <https://github.com/TektoNomine/integer-array-manager-cpp>

**Personal Reflection – Dylan Ryan Gleeson**

**Personal Reflection – Dylan Ryan Gleeson**