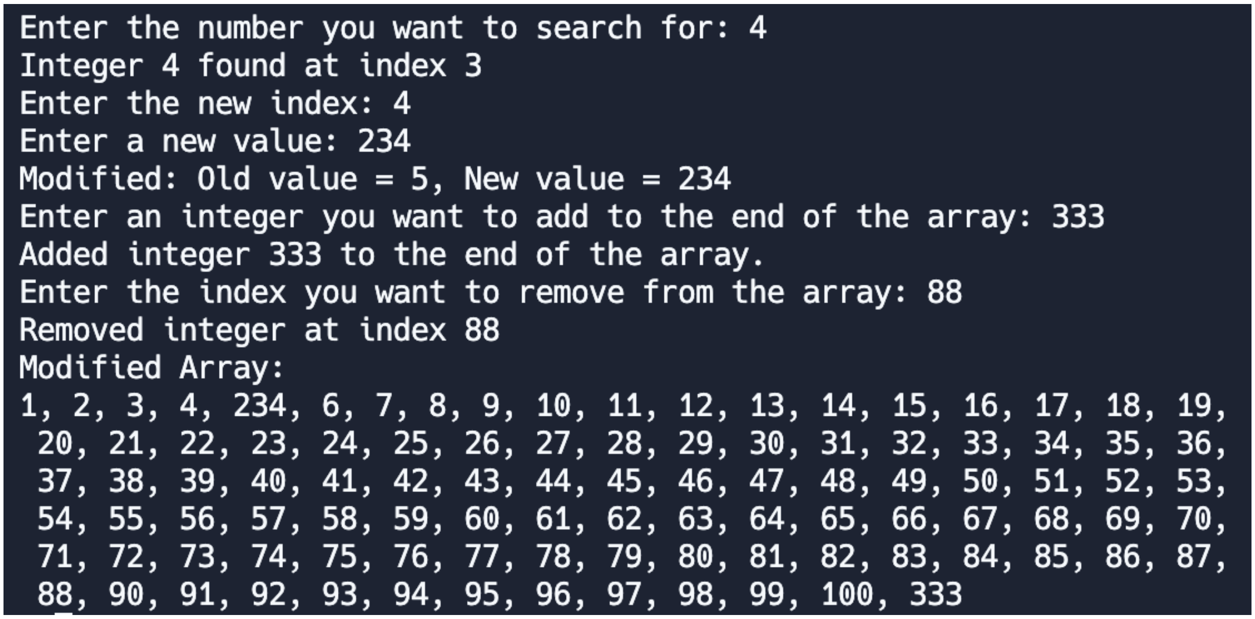
Abby Gerstner

How to run this file:

1. Open main.h, main1.cpp, and array.txt.
   - Array.txt is a file that has 100 integers listed 1-100
   - Main.h is the header file where all class function declarations are made.
   - Main1.cpp is where the functions and implementations are made.
2. Press run.
3. Type the integer you want to find into the array and press enter.
4. The output will be "Integer __ found at index __" if input is valid. If not, an error message will occur and stop the program.
5. Type the index that you want to insert a new integer and press enter.
6. Type the integer you want inserted at that index and press enter.
7. The output will be "Modified: Old value = __, New value = __" if input is valid. If not, an error message will occur and stop the program.
8. Type the an integer you want to add to the end of the array and press enter.
9. The output will be "Added integer __ to the end of the array" if input is valid. If not, an error message will occur and stop the program.
10. Type the index you want to remove from the array and press enter.
11. The output will be "Removed integer at index __" if input is valid. If not, an error message will occur and stop the program.
12. The program will then print the modified array to the console and the program will stop.

Here is the output:



```
Enter the number you want to search for: 4
Integer 4 found at index 3
Enter the new index: 4
Enter a new value: 234
Modified: Old value = 5, New value = 234
Enter an integer you want to add to the end of the array: 333
Added integer 333 to the end of the array.
Enter the index you want to remove from the array: 88
Removed integer at index 88
Modified Array:
1, 2, 3, 4, 234, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87,
 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 333
```

All assumptions:

File Input Assumptions:

The program assumes that there is an input file named "array.txt" in the same directory as the executable.

The input file should contain a sequence of integers separated by whitespace, and the program reads and processes these integers.

Data Type Assumptions:
The program assumes that the integers read from the input file and entered by the user are of type int. It doesn't handle non-integer input gracefully.

Error Handling Assumptions:
The code assumes that errors related to file opening, memory allocation, and invalid user input will be handled by exiting the program (exit(1)). It doesn't provide a mechanism for recovering from these errors.
It also assumes that input validation is the responsibility of the user and doesn't include extensive error-checking for invalid inputs.

Memory Management Assumptions:
The code assumes that the user is responsible for deleting the dynamically allocated memory (delete[] dataArray) to avoid memory leaks.

Indexing Assumptions:
Functions like modifyValue and replaceOrRemove assume that the user provides valid indices within the bounds of the array.
No bounds checking is performed in these functions.

Console Output Assumptions:
The code assumes that the console is available for input and output. It doesn't provide a graphical user interface.

User Interaction Assumptions:
The code assumes that the user will interact with the program through the console and follow the prompts for entering values and indices.

Function Usage Assumptions:
The code assumes that functions like modifyValue, replaceOrRemove, and others will be used correctly with valid inputs.
It does not anticipate or gracefully handle misuse of these functions.

Positive Integers:
The code doesn't explicitly check for negative integers or zero. It assumes that the user provides positive integers for array manipulation.

No Concurrent Access:
The code does not handle concurrent access to the array or synchronization between multiple threads. It assumes single-threaded execution.