Assignment #5 - Array Practice

Due: Tuesday, November 6th 11:59:00pm

Objective

To gain practice with arrays and common array algorithms, as well as the use of array parameters in functions.

Task

This assignment will consist of writing several functions that manipulate arrays or access data from arrays, as well as a test program that will allow interactive testing of the array functions.

Part 1: Functions

Write the following functions. Each one takes in an integer array as a parameter, and other necessary parameters and returns are described - read carefully. **Make sure the parameters are in the order specified.** **Make sure to use the `const` qualifier on the array parameter on any function where it is *appropriate*.** A sample CALL is given for each function.

- **FillArray**

  Write a function called `FillArray` that takes in four parameters:
  - an integer array
  - size of the array
  - a min value (int)
  - a max value (int)

  This function should fill the arrays contents with random values within the range min-->max (inclusive of min and max) This function should not return any value. For instance:

  ```
  The call:
  FillArray(list,SIZE,-50,100);
  would fill the array list with random numbers within the range -50 to 100.
  ```

- **Insert**

  Write a function called `Insert` that takes in four parameters:
  - an integer array
  - the size of the array
  - the new value to be inserted into the array
  - the index at which to insert the new value

  This function should insert a a new value into the array, at the specified index. Note that this means that

other values have to "move" to make room. The last value in the array will just disappear from the array. If the index is out of bounds for the array, abort the function with no change made to the array. This function does not return a value. Sample call:

```
// Suppose the array "list" is {2, 4, 6, 8, 10, 12}

Insert(list, 6, 100, 3);      // insert the value 100 at index 3.

// "list" is now {2, 4, 6, 100, 8, 10}
```

- **Delete**

  Write a function called `Delete` that takes in three parameters:
  - an integer array
  - the size of the array
  - the index of the item to delete

  This function should delete the value at the given index. The remaining items in the array will need to shift over to fill the "empty" slot. The last item in the array (now vacant) should be set to 0. If the given index is out of bounds for the array, abort the function without deleting anything. This function does not return a value. Sample call:

```
// Suppose the array "list" is {2, 4, 6, 8, 10, 12}

Delete(list, 6, 2);   // delete the value at index 2.

// "list" is now {2, 4, 8, 10, 12, 0}
```

- **Reverse**

  Write a function called `Reverse` that takes in two parameters:
  - an integer array
  - the size of the array
  - YOU MUST DO THIS TASK WTIHOUT CREATING ANOTHER ARRAY. Aka, you have to move the items in place within the original array.

  This function should reverse the order of the array's contents. No returned value. Sample call:

```
// Suppose the array "list" is {2, 4, 6, 8, 10, 12}

Reverse(list, 6);     // Reverse the array "list"

// "list" is now {12, 10, 8, 6, 4, 2}
```

- **Largest Odd Number**

  Write a function called `MaxOdd` that takes in two parameters:
  - an integer array
  - the size of the array

  The function should look at the array and determine the maximum (highest) odd number that is in the array. (You may assume that there is at least one odd number in the array). The function should return

the max odd value it finds as an integer.

```
// Suppose "list" is {2, 5, 5, 10, 5, 10, 3, 4, 5, 9, -1, 6}

// The call to the function:
      MaxEven(list,SIZE); //would return the value 9
```

Note that none of the five functions you'll write above do any keyboard input or screen output. You're also not permitted to create any other arrays in your code besides the original one in main().

Part 2: Test your Functions with your main() function

To help you test and get you started, I've provided you with a STARTER FILE which you SHOULD use to start. Fill in your code into this starter file. The starter file already contains the `PrintArray` function that we looked at in lecture class. You can use this in your main function which will test your array functions, as well as anywhere else you need a PrintArray functionality.

Write a `main()` function that creates an array of size `SIZE` (a constant given in the starter file). Since this is a constant, it can be changed for testing different sizes of arrays easily. Use this constant whenever referring to the array's size from `main()` (instead of using a hard-coded literal value).

Then, the program should go into a menu loop, presenting the user with the following menu the first time:

```
        ** Given features **
P       Print the array contents

        ** Function Tests **
F       Fill the array with random values
I       Insert
D       Delete
R       Reverse
X       Max Odd Value


M       Print this menu
Q       Quit this program
```

- The menu only needs to be printed explicitly the first time. Then, only re-print the menu if the user selects the **M** menu option.
- Prompt the user the first time and after every user selection to enter their next menu selection with the following:

  ```
  Enter your menu selection:
  ```

- The first choice should just invoke the two already given function: `PrintArray`.
- The **Q** option should exit the menu loop and allow the program to end. Make sure to print out the final array before quitting the program
- The 5 menu options under `**Function Tests**` will test your functions. Some of them call for extra user input. The options should behave as follows (make sure to do any user input in the order specified):

- **F**: The option F will call the `FillArray` function. You may assume this option is selected first by the user to populate the array with data before any other options are selected. When the user selects this option, prompt the user for the minimum random value and the maximum random value. You may assume they give legitimate minimum and maximum values. Then call your `FillArray` function passing in these user entered min and max values to fill up your array with data.
- **I:** Prompt the user to type the value to be inserted, then the index to insert at. Call the `Insert` function appropriately, then print out the contents of the array
- **D:** Prompt the user to type the index to be deleted. Call the `Delete` function appropriately, then print out the contents of the array.
- **R:** No extra input required. Just call the `Reverse` function appropriately, then print out the array
- **X:** Call the `MaxOdd` function and print out a message indicating the largest odd number found Example:

```
The maximum odd value in the array is 11
```

Hint: A good way to implement a menu loop is a switch statement (for processing a menu selection) inside of a `do-while` loop (for repeating the menu selection process).

General Requirements

- ***No global variables***
- The required array tasks **must** be performed with the **functions** specified (not just with a single main() routine). **Note that your 5 required functions should not have any cout/cin statements in them.** You can have additional functions if you like, however make sure to have the 5 required ones, and that your required functions adhere to the specs.
- **You may not change the provided function in any way** (`PrintArray`)
- You are not permitted to create any other arrays in your code besides the original one you create in main()
- You will need the `iostream` library. You may also use `iomanip` and/or `cctype`, `ctime`, and `cstdlib` if you like. (No other libraries).
- As always, your source code must be readable and appropriately documented

---

SAMPLE RUNS

SAMPLE RUN HERE This should give you an idea of how your program should behave. So long as the output formatting of the menu is clean, and the overall formatting of your output looks like this, you'll be fine. NOTE: Not all scenarios are shown - there's lots! So make sure you read through the assignment carefully to see how to handle certain scenarios.

Submitting:

Submit your assignment using the command below.

```
~vastola/usub/submit1 arrays.cpp
```