CS 162
Programming Lab 4

For this exercise, you will create a dynamic array and manipulate it using pointers.

Problem Description

While the program will be an introduction to pointers and dynamic memory, it is also a review of functions, arrays, searching, and sorting.

For this program you will get a value from the user between 10 and 20, create an array of this size using dynamic memory allocation, fill the array with random numbers between 1 and 99, sort the array, and then display the array. You will then ask the user for 3 numbers and for each value you will say whether it is in the array or not.

In order to give you more practice with pointers, you are to do all array element accesses with a pointer instead of using subscripting. (int * ptr = array; *ptr = value; instead of array[0] = value;)

Required functions

getSize -- This function will ask the user for a value between 10 and 20, verify that they entered an integer in this range, and return it to main.

createArray – This function is passed the length of the desired array. It will create an array of that length using dynamic memory allocation (the new operator) and fill it with random numbers between 1 and 99. For all memory accesses you are to use pointer dereferencing (*array) instead of array subscripting (array[0]). Once the array is created and filled, you will return it using a pointer to an int.

sortArray – The next function is a sort function. You will pass in the array and its length and then sort it using **one of the sorts** covered in the module on searching and sorting. The array should be sorted in ascending order (smallest at first location, largest at last location).

displayArray – The fourth function is a display function. You will pass the array to the function along with its length. Then you will display the values, five per line. You should use the setw operator to align your numbers in neat columns. Again, you are to do all your memory references using pointer dereferencing.

Search – The fifth function performs a **binary search**. You will pass in the array and its length to the function along with a value. It should return true if the value is in the array, false otherwise. You are to use a **binary search** for this function. You can make this recursive if you would like, but it is not required.

getInteger – This function is like getSize, but asks the user for a number between 1 and 99, validates it, and returns it to the calling program.

Program Requirements

You need to use **constants** where appropriate and **comment** all functions.

You should free up the memory from your array, using **delete** properly when you are done.

Program Suggestion

Your main should have this basic structure:

```
getSize
createArray
sortArray
displayArray
loop three times
    getInteger
    search
    output if found or not
```

Define your program with stub functions, then code and test functions in this order:

1. getSize
2. createArray
3. displayArray
4. sortArray
5. search

You could design a single function that is passed a range (min to max), asks the user for a number in that range, validates it, and returns it to the calling program. This could then be used for getSize and getInteger.