Programming Assignment #3

Practice with pointers and dynamic memory allocation

CS 2308.003, Fall 2011
Instructor: Jill Seaman

Due: in class **Thursday, 9/29/2011** (upload by **10:00am Thursday 9/29/2011**)

Problem:

Write a C++ program that will test five functions described below that use pointers and dynamic memory allocation.

Input: None!

The Functions:

You will write the five functions described below. Then you will call them from the main function, to demonstrate their correctness (or usefulness, in #5).

1. **minimum**: takes an int array and the array's size as arguments. It should return the minimum value of the array elements. Do not use square brackets ANYWHERE in the function (use pointers instead). Bonus: Do not use the loop variable in the body of the loop.

2. **swapTimesTen**: The following function uses reference parameters. Rewrite the function so it uses pointers instead of reference variables. When you test this function from the main program, demonstrate that it changes the values of the variables passed into it.

```
int swapTimesTen (int &x, int &y)
{
   int temp = x;
   x = y * 10;
   y = temp * 10;
   return x + y;
}
```

3. **expand**: takes an int array and the array's size as arguments. It should create a new array that is twice the size of the argument array. The function should copy the contents of the argument array to the new array, and initialize the unused elements of the second array with 0. The function should return a pointer to the new array.

4. **concatenate**: takes two int arrays and the arrays' sizes as arguments (that's 4 arguments). It should create a new array big enough to store both arrays. Then it should copy the contents of the first array to the new array, and then copy the contents of the second array to the new array in the remaining elements.

5. **duplicateArray**: takes an int array, and the array's size as arguments. This function creates a new array that is a duplicate of the argument array and returns a pointer to the new array (please copy from bottom of page 542). In the main function, call this function as is to make a "sub-array" of another. Initialize an int array named (let's call it aa) with several values. Use a call to duplicateArray to return a new array containing only elements aa[5], aa[6], aa[7], and aa[8]. You will be passing something other than the array aa and its size.

Output:

Test these five functions using main as a "driver": a driver is a program (or function) that tests other functions by calling them (perhaps repeatedly). The driver passes test data as arguments to the functions. If the function returns data, the driver can display it on the screen, or compare it to expected result data. This allows you to see how the function performs in isolation from the rest of the program it will eventually be a part of.

Your main function should be a series of five sections, one per function being tested above. You should select appropriate test data for each function and then call that function using the test data. For each function, you should output three lines: a label indicating which function is being tested, the expected results (use one string with the values "hard coded" into it), and the actual results (use the actual values returned/altered by the function in this line).

```
testing minimum:
Expected minimum: -8
Actual minimum:   -8

testing swapTimesTen
Expected result: 80  a: 50  b: 30
Actual results : 80  a: 50  b: 30

testing expand:
Expected result: 1 2 3 4 5 6 7 8 9 0 0 0 0 0 0 0 0 0 0 0
Actual result:   1 2 3 4 5 6 7 8 9 0 0 0 0 0 0 0 0 0 0 0

testing concat:
Expected result: 1 2 3 4 5 6 7 8 9 0 11 22 33 44 55
Actual result:   1 2 3 4 5 6 7 8 9 0 11 22 33 44 55

test using duplicateArray:
Expected result: 6 7 8 9
Actual result:   6 7 8 9
```

NOTES:

- This program does not need to be done in a Unix environment. You may use whatever C++ programming environment (Visual C++, Dev-C++, etc.) you prefer.

- DO NOT change the names of the functions!

- You do not need to use named constants for your test data (or array sizes) in this assignment.

- You DO need to follow the rest of the style guidelines described here: http://www.cs.txstate.edu/~js236/styleguidelines.txt
  (there is a link to these on the course webpage as well).

- Your program should release any dynamically allocated memory when it is finished using it.

- Optional: you may want a function that displays the values of an int array on one line, separated by spaces.

Logistics:

Name your file **assign3_xxxxxxxxx.cpp** where xxxxxxxxx is your 9 character TX state ID number, the one that is on your ID card. It should look something like this: A04123456. If yours is just six digits, then add "A00" to the front.

There are **two** steps to the turn-in process:

1. Submit an **electronic copy** using the following upload link:
   http://www.cs.txstate.edu/~js236/homework
   (the link is also on the class webpage).
   Click on CS2308.003, and log in with your Net ID and follow the directions to upload your file.

2. Submit a **printout** of the file at the beginning of class when the assignment is due. Please print your name on the front page (stapled together, please, if you have more than one page of output).