```
// File Name: assign3_a_s1316.cpp
//
// Author: Archana Sridhar
// Net ID: a_s1316
// Date: 3/04/2019
// Assignment Number: 3
// CS 5301 Spring 2019
// Instructor: Dr.Jill Seaman
//
// C++ program that will implement and test the five functions that use pointers and dynamic
memory allocation.

#include <iostream>
using namespace std;

//*************************************************************************************
// minimum: determines the minimum in the array of elements
// array : the array of integer elements
// size : the number of elements in the array
// returns the minimum in the array of elements
//*************************************************************************************
int minimum(int *array, int size)
{
    int min = *array;
    for(int i = 1; i < size ; i++)
    {
        if (*(array+i) < min)
        {
            min=*(array+i);
        }
    }
    return min;
}
//*************************************************************************************
// swapTimesTen: demonstrate that it swaps the values of the variables passed into it
// x : represents an integer variable
// y : represents an integer variable
// returns the sum of the variables
//*************************************************************************************
int swapTimesTen (int *x, int *y)
{
    int temp = *x;
    *x = *y * 10;
    *y = temp * 10;
    return *x + *y;
}
//*************************************************************************************
// doubleArray: create a new array that is twice the size of the argument array
// array : the array of integer elements
// size : the number of elements in the array
// returns the new array that copies the contents of the original array in the first
//         half and contents multiplied by 2 in the second half of the array
//*************************************************************************************
int *doubleArray (int *array, int size)
{
    if (size < 0)
        return NULL;

    int *newArray = new int[size*2];
    for (int i = 0 ; i < size ; i++)
    {
        *(newArray+i) = *(array+i);
        *(newArray+size+i) = 2*(*(array+i));
    }
```

```
    return newArray;
}
//*********************************************************************************
// subArray : creates a new array that is a copy of the elements from the original array
//           starting at the start index, and has length equal to the length argument
// array : the array of integer elements
// start : the start position of the array element from which the elements
//         should be copied
// length : the length to which the elements needs to be copied to the new array
// returns the copy of the elements from original array into a new Array
//*********************************************************************************
int *subArray (int *array, int start, int length)
{
    if (length < 0)
        return NULL;

    int *newArray = new int[length];
    for (int i = 0; i < length; i++)
    {
        *(newArray+i) = *(array+start+i);
    }
    return newArray;
}
//*********************************************************************************
// duplicateArray: creates a new array that is a copy of elemnents from original array
// array : the array of integer elements
// size : the number of elements in the array
// returns the new array that is a copy of elements from original array
//*********************************************************************************
int *duplicateArray (int *array, int size)
{
    if (size <= 0)
        return NULL;

    int *newArray = new int [size];
    for (int i = 0; i < size; i++)
    {
        *(newArray+i) = *(array+i);
    }
    return newArray;
}

//*********************************************************************************
// subArray2 : creates a new array that is a copy of of elemnents from original array
// array : the array of integer elements
// start : the start position of the array element from which the elements
//         should be copied
// length : the length to which the elements needs to be copied to the new array
// returns the part in the array of elements
//*********************************************************************************
int *subArray2 (int *array, int start, int length)
{
    int *result = duplicateArray(array+start,length);
    return result;
}
//*********************************************************************************
// testData: displays the test data (array of elements initialized)
// array : the array of integer elements
// size : the number of elements in the array
// returns the array of elements
//*********************************************************************************
void testData(int *array,int size)
{
    for (int i = 0; i < size ; i++)
        cout << *(array+i) << " ";
```

```
}
//*********************************************************************************
// displayResults : outputs the array of elements
// array : the array of integer elements
// size : the number of elements in the array
// returns the array of elements
//*********************************************************************************
void displayResults(int *array, int size)
{
    for (int i = 0; i < size ; i++)
        cout << *(array+i) << " ";
}
int main()
{
    //calling minimum function
    cout << "testing minimum" << endl;
    int arr[10] = {1, 2, 3, 4, 5, 0, -7, 8, 9, 10};
    cout << "test data : ";
    testData(arr, 10);
    cout << endl;
    cout << "Expected minimum : -7" <<endl;
    int min = minimum(arr, 10);
    cout << "Actual minimum   : " << min << endl;
    int arr1[10] = {1, 2, 3, 4, 5, 16, 7, 8, 9, 0};
    cout <<"test data : ";
    testData(arr1, 10);
    cout << endl;
    cout << "Expected minimum : 0" <<endl;
    int min1 = minimum(arr1, 10);
    cout << "Actual minimum   : " << min1 << endl;
    cout << endl;

    //calling swapTimesTen function
    int a = 3, b = 5;
    cout << "testing swapTimesTen" << endl;
    cout << "Expected Result :80  a:50  b:30" << endl;
    int c = swapTimesTen(&a, &b);
    cout << "Actual Results  :" << c <<"  a:" <<a <<"  b:"<<b <<endl;
    cout << endl;

    //calling doubleArray fucntion
    cout << "testing doubleArray" << endl;
    int arr2[9]={1, 2, 3, 4, 5, 6, 7, 8, 9};
    cout << "test data : ";
    testData(arr2, 9);
    cout << endl;
    int *arr3 = doubleArray(arr2,9);
    cout << "Expected result : 1 2 3 4 5 6 7 8 9 2 4 6 8 10 12 14 16 18" << endl;
    cout << "Actual result   : ";
    displayResults(arr3, 18);
    cout << endl << endl;
    delete [] arr3;

    //calling subArray function
    cout << "testing subArray" << endl;
    int arr4[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    cout << "test data : " ;
    testData(arr4,10);
    cout << endl;
    int *arr5 = subArray(arr4,5,4);
    cout << "start: 5 length: 4" << endl;
    cout << "Expected result: 6 7 8 9" << endl;
    cout << "Actual result  : ";
    displayResults(arr5, 4);
    cout << endl << endl;
```

```
    delete [] arr5;

    //calling subArray2 function
    cout << "testing subArray2" << endl;
    int arr6[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    cout << "test data : ";
    testData(arr6, 10);
    cout << endl;
    int *arr7 = subArray2(arr6,5,4);
    cout << "start: 5 length: 4" << endl;
    cout << "Expected result: 6 7 8 9" << endl;
    cout << "Actual result  : ";
    displayResults(arr7, 4);
    cout << endl;
    delete [] arr7;

    return 0;
}
```