COMP 53: Arrays and Vectors Lab, part 2

Instructions: In this lab, we are going to review vectors.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code files list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **37 points** in aggregate. The details are given in the following.

1 `main.cpp`

In `main.cpp` do the following step by step:

1. Globally define array `a[]` of integers consisting of values: 5, 7,-2, 8, 11, -9, 4, 6, 12, and -1 in order.
2. Globally define array `b[]` of integers consisting of values: 4,16,9,-2, 1, 1, -2, 9, 16, and 4 in order.
3. Globally define three vectors of integers, without initial values. Call them `vec1`, `vec2`, and `vec3` ***(3 points)***.
4. Define the following functions on vectors. In all of the following functions pass the vector by reference.
   (a) Define function `void initVector(...)` that receives a vector of integers as its first input, an array of integers as a second input, and an integer as its third input. The third input represents the number of elements in the input array. Initialize the input vector with the elements existing in the input array ***(3 points)***.
   (b) Define function `void printVector(...)` that receives a vector of integers as input and prints the elements residing within that vector in the standard output. Since the input vector is not being modified by this function, define it to be constant ***(3 points)***.
   (c) Define function `int minVector(...)` that receives a vector of integers as input and returns the least element. Since the input vector is not being modified by this function, define it to be constant ***(3 points)***.
   (d) Define function `int productVector(...)` that receives a vector of integers as input and returns the product of all elements within the input vector. Since the input vector is not being modified by this function, define it to be constant ***(3 points)***.
   (e) Define function `void copyVector(...)` that receives two vectors of integers as input and copies the first vector into the second vector element by element, i.e., second vector becomes identical to the first one. Do not use vector traversal through loops for this purpose. Since the first input vector is not being modified by this function, define it to be constant ***(3 points)***.
   (f) Define function `void greaterVector(...)` that receives three vectors of integers as input, compares the first two vectors element by element, and puts the larger one in the third vector (in the same index). Since the first two input vectors are not being modified by this function, define them to be constant ***(3 points)***.
   (g) Define function `bool isPalindrome(...)` that receives a vector of integers as input and returns true if the input vector is a palindrome. Otherwise it returns false. A sequence of items is a palindrome if it reads the same backward as forward. For example, the vector of numbers 1,2,3,2,1 and the vector of numbers 1,2,3,3,2,1 are palindromes. However, the vector of numbers 1,2,3,1 is not a palindrome. Since the input vector is not being modified by this function, define it to be constant ***(3 points)***.
   (h) Define function `void updateLast(...)` that receives a vector of integers as input along with an integer. The function is supposed to change the last item of the vector to the integer input. Do not use `at()` function for this purpose ***(3 points)***.

In `main()` function do the following step by step, using the functions defined above:

(i) Print out the initial size of `vec1` ***(1 points)***.

(ii) Initialize `vec1` according to array `a[]` ***(1 points)***.

(iii) Print out the size of `vec1` after initialization according to the function defined above ***(1 points)***.

(iv) Print out the elements of `vec1`, according to the function defined above ***(1 points)***.

(v) Initialize `vec2` according to array `b[]` ***(1 points)***.

(vi) Print out the elements of `vec2`, according to the function defined above ***(1 points)***.

(vii) Print out the minimum element of `vec1` ***(1 points)***.

(viii) Print out the product of all elements of `vec1` ***(1 points)***.

(ix) Copy `vec1` to `vec3` according to the function defined above, and print out `vec3`'s elements ***(1 points)***.

(x) Collect larger elements from `vec1` and `vec2` to `vec3`. Next, print out `vec3`'s elements ***(1 points)***.

(xi) Check if `vec1` is a palindrome according to the function defined above, and report the result ***(1 points)***.

(xii) Check if `vec2` is a palindrome according to the function defined above, and report the result ***(1 points)***.

(xiii) Update the last element of `vec3` to `7`, according to the function defined above ***(1 points)***.

The output of the program may look like the following:

```
initial size of vec1: 0
size of vec1 after initialization: 10
vec1 content: 5, 7, -2, 8, 11, -9, 4, 6, 12, -1
vec2 content: 4, 16, 9, -2, 1, 1, -2, 9, 16, 4
minimum of vec1: -9
product of vec1: -15966720
copy vec1 to vec3: 5, 7, -2, 8, 11, -9, 4, 6, 12, -1
collect larger elements from vec1 and vec2 to vec3: 5, 16, 9, 8, 11, 1, 4, 9, 16, 4
vec1 is not Palindrome
vec2 is Palindrome
Updating the last element of vec3: 5, 16, 9, 8, 11, 1, 4, 9, 16, 7
```