COMP 53: Containers Lab, part 1

Instructions: In this lab, we are going to review range-based `for` loops.

- 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 **25 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 a vector of integers, without initial values. Call it `vec1`.
3. Define the following functions on vectors. In all of the following functions pass the vector by reference. In addition, you must use **range-based `for` loops** for all required iterations overs vectors.
   (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 doubleVector(...)` that receives a vector of integers as input and doubles every element in that vector. Note that the vector is being modified by this function ***(3 points)***.
   (f) Define function `void aggregationVector(...)` that receives a vector of integers as input and computes an aggregated vector defined as follows:
   - First element of the aggregated vector is the same as the first element of the input vector.
   - Second element of the aggregated vector is the summation of the first two elements of the input vector.
   - Third element of the aggregated vector is the summation of the first three elements of the input vector.
   - ...

   Note that the vector is being modified by this function ***(4 points)***.

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

(i) Initialize `vec1` according to array `a[]` using the function defined above ***(1 points)***.

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

(iii) Print out the minimum element of `vec1` using the functions defined above ***(1 points)***.

(iv) Print out the product of all elements of `vec1` using the functions defined above ***(1 points)***.

(v) Print out the updated `vec1` after doubling each element (using the function defined above) ***(1 points)***.

(vi) Print out the updated `vec1` after aggregating its element values (using the function defined above) ***(1 points)***.

The output of the program may look like the following:

```
vec1 content: 5, 7, -2, 8, 11, -9, 4, 6, 12, -1
minimum of vec1: -9
product of vec1: -15966720
doubling each element of vec1: 10, 14, -4, 16, 22, -18, 8, 12, 24, -2
aggregating element values in vec1: 10, 24, 20, 36, 58, 40, 48, 60, 84, 82
```