Lab Exercises

Exercise 1

Print 100 random numbers. Each random number must be obtained by calling a lambda function that generates a random number between 1 and 100.

Exercise 2

Use `std::sort` to sort (in ascending order) a vector of integers containing the elements `{1, 3, 8, 6, 4, 5, 7, 2, 0, 9}`. After sorting is completed, print the number of comparisons that were needed during sorting.

Hint: Write a lambda function that compare two **`int`**s and also increment the captured `count` variable.

Exercise 3

(a) Write a lambda function that takes an integer and returns **`true`** if the integer is odd, otherwise it returns **`false`**.

(b) Given an object `v` of type `std::vector<int>`, write code to determine whether all of the elements of the vector are odd integers. Use a lambda function from part (a) and `std::all_of` algorithm.

Exercise 4

(a) Write a lambda function that takes two vectors of integers and returns **`true`** if kth element of the first vector is greater than the kth element of the second vector. Otherwise, it returns **`false`**. (where k is a given integer captured by the lambda function)

(b) The *Object Oriented Programming Techniques* class has `m` students and `n` exams. You are given a `m x n` 2D integer vector `score`, where each row represents one student and `score[i][j]` denotes the score the ith student got in the jth exam. The matrix `score` contains *distinct* integers only.

You are also given an integer `k`. Write a function with following signature:

```
void sortStudents(vector<vector<int>>& score, int k) {
}
```

which sort the students (i.e., the rows of the matrix) by their scores in the kth (0-indexed) exam from the highest to the lowest. Use the lambda function you wrote in part (a) to compare the scores of the students and `std::sort` to sort the students.

Source: `https://leetcode.com/problems/sort-the-students-by-their-kth-score/description/`

Exercise 5

(a) Write a lambda function that takes an integer and returns $a * x * x + b * x + c$ where a, b, and c are captured values.

(b) Write a function called `transform_quadratic` that matches the following signature:

```
void transform_quadratic(vector<int>& v, double a, double b, double c) {
}
```

The function should replace each element `x` in the vector `v` with the value `a*x*x + b*x + c`. Use `std::transform` and lambda function from part (a) to apply the transformation.