Recursion Project

Objectives

The objective of this lab is for you to become more comfortable with recursion. Please keep in mind that none of the problems that you have been asked to solve in this lab should be implemented recursively. However, these problems are easy to understand and solve iteratively and as a result, they you should be able to focus on recursive thinking and not be inhibited by the complexity of the problems.

Introduction

For this project, you have been asked to write solutions to 4 independent problems. For each problem, you will have to write an iterative solution and a recursive solution. Here are the rules that apply to all these problems.

1. In your iterative solutions, you may not write nested loops in any single function. Rather, if you have nested loops, you will have to write a new function to perform the task of the inner loop. Using helper functions of this type, when appropriate, add clarity to your solutions and as a result, you might be better able to come up with recursive solutions to the same problems.
2. In your recursive solutions, you may not use any loops. If you were to write helper functions for your recursive solutions, they too, have to be recursive.
3. In either of the iterative and the recursive solutions, you will have to design the function prototypes for your helper functions.
4. The input file for each problem contains a set of integers. Therefore, you should write a function that, given a file name and an int vector, opens the file, reads the stores the input values into the vector. Below is the function-prototype for this function. You call this function in your main function and then use the vector into which the numbers are stored by read_numbers to call other functions.

   ```
   void read_numbers(std::string fileName, std::vector<int> &v);
   ```
5. All these function prototypes, the ones that we have provided (see below and the descriptions of each of the four problems) and those that you come up with for your helper functions should be stored in a file called project2.hpp. This include the prototype for read_numbers. The solutions for these functions get stored in project2.cpp. These functions are all global functions -- they are not part of any C++ class. That is, you do not create any C++ classes for this project.

6. Based on the previous item, your solution to this project should consist of 3 files; project2.hpp, project2.cpp, and main.cpp. In particular, main.cpp includes project2.hpp and only contains the definition of function main -- *int main(int argc, char *argv[])*;.
7. Each problem has a base name. The names of the iterative and the recursive functions for any of the problems use, as a prefix, this base name. For example, the base name for Count Inversions problem is count_inversions. Based on that, the names of the iterative and recursive functions for the problem are count_inversions_iterative and count_inversions _recursive, respectively. Please note that the letter are all lower-case.
8. Therefore, the following is a list of function prototypes that get stored in project2.hpp and their implementations get stored in project2.cpp. You add the function prototypes of the helper function that you use to this file and to this list.
   ```
   int count_inversions_iterative(const std::vector<int>
   &numbers);
   int count_inversions_recursive(const std::vector<int>
   &numbers, int n);
   void largest_left_iterative(const std::vector<int>
   &numbers, std::vector<int> &result);
   void largest_left_recursive(const std::vector<int>
   &numbers, std::vector<int> &result, int n);
   void larger_left_right_iterative(const std::vector<int>
   &numbers, std::vector<int> &result);
   void larger_left_right_recursive(const std::vector<int>
   &numbers, std::vector<int> &result, int n);
   int increasing_sequences_iterative(std::vector<int>
   &numbers);
   int increasing_sequences_recursive(std::vector<int>
   &numbers, int startIdx);
   void read_numbers(std::string fileName, std::vector<int>
   &v);
   ```
9. You probably want to write your main function in such a way to test one of these problems at a time. For example, to test count inversions, in your main function, you would comment out the code segments that test other functions, except for read_input, and then, add the code that calls count_inversions_iterative and/or count_inversions_recursive.
10. Therefore, you (and us) should be able to compile and run your count_inversion_iterative in the following way. Note that, from the way that we compile and run the program, it is not obvious which function is being tested. But, of course, as you develop your code, you know which function you are

testing. We will run your code against our own main functions to test your solution to each of the problems. Please note that in spite of the fact that we use our own main functions, you should provide a main function that you have used to test your solutions to these functions. main.cpp includes project2.hpp.

- g++ -std=c++17 project2.cpp main.cpp
- ./a.out count_inversions_input.txt