CSE 3150 — C++ Essentials — Yufeng Wu — Fall 2024
Programming Assignment 4. Due: 11:59 pm @ 09/27/2024.

There are objectives in this assignments.

1. Get familar with pointer and reference.
2. Basic flow control (loop, etc)

1 Pointer and reference

The purpose of this assignment is practicing pointers and references. Here, you are given an array of pointers to strings. You need to implement a function called RemoveDupPointers that takes this array of pointers and remove the pointers that have duplicate strings. Note: you should test whether two pointers point to identical strings, not whether two pointers have the same address. Also note: don't change relative order of the pointers that are kept.

For example, suppose you are given three strings:
string s1="abc", s2 = "bcd", s3="abc";
The array of pointers: array[] = {&s1, &s2, &s3, &s1};
You invoke: RemoveDupPointers(array).
Then the array should only contain: array[] = {&s1, &s2}

Note: I will use C++ STL vector to represent an array. Essentially, STL vector is a "better" array. I want you to start to experiment with this powerful utility class. Google it to find references on how to use STL vector. Briefly, you can access a vector by subscripting, e.g. *vec*[10] would give you the element of *vec* (a vector) at index 10. You can find out the size of vector by invoking like *vec.size*() (which returns the number of elements iin the vector).

2 Runs in string

You are given a string *str*. You need to implement a function that finds the maximal runs in the string. A run is defined as a maximal consecutive segment of identical characters in the string. Maximal means you cannot extend the segment to the left or to the right. For example, suppose you are given: if the string is 1337772211. Then there are five runs: 1; 33; 7 7 7; 2 2; and 1 1.

You need to return the number of runs in the string and also return the starting positions of runs in the passsed-in array. You may assume the array is large enough. You don't need to set values in this array that are not in the valid range of number of runs.

3 Finding common number in two arrays

You are given two (unordered) arrays A and B. You want to find the smallest common number in A and B. For this, you are implement a function called *ECCommonNumber*, which takes five parameters and return a boolean (true if there is a common number) and false otherwise. The five parameters are (from the first to the last):

1. A: the first array (of integers)
2. sz_A: size of the first array
3. B: the second array (of integers)
4. sz_B: size of the second array

5. *val*: the (smallest) common integer found. Note: this number should be part of returned value to the caller. That is, after calling ECCommonNumber, *val* will be the found common integer. If there is no common integer, *val* can be of any value.

The following lists things to consider when you implement this function.

1. First, you should define the function prototype (i.e, the signature of the fuction: the types of the parameters). Note: the two arrays cannot be changed by your function (we will have test to ensure that). This is because usually the caller doesn't expect the arrays will change (so called side effect) after calling the function.

2. You must implement the function as **efficient** as possible. Recall that efficiency is one of the most important aspects of C++ programming. You should carefully think about the algorithm you are to use. The most straightforward algorithm is simply comparing each pair of numbers in the two lists. However, this is very slow when the arrays get large. We will have run time check to ensure your code is fast when the sizes of array increase. Hint: I would suggest using binary search on sorted array; do you see why this is faster?

3. Don't reinvent the wheel! If you can, try to use the provided standard library functions. These include:

   (a) Sort: C++ has a function for sorting. To sort an array *arr*, you can simply call: std::sort(arr, arr+sz); where *sz* is the size of the array. Try it yourself!

   (b) Binary search: C++ has the binary search implemented for you. Google to see how it works.

4 Instructions for submission

Same as the last assignment.

About the filenames

For almost all programs for this course, I append a prefix "EC" which stands for Essentials of C++ (the course name). This is to avoid any potential naming conflicts, and also serve as a note that the code you wrote is for this course.