CSCI 340 - Data Structures and Algorithm Analysis Data Structure: Vector Programming Focus: Exposure to the Standard Template Library (STL) and review of C++

Two Search Algorithms

For this computer assignment, you are to write and implement a C++ program that uses two search algorithms (linear search and binary search) on randomly generated integers stored in a vector from STL.

Put the definitions of all constants and the prototypes of your subroutines in your header file twosearch.h, and complete the implementation of the `main()` routine in your source file twosearch.cc, along with the implementations of of your subroutines, as described below.

Do the following in your **`main()`** routine:

1) Define two vectors (`A` and `B`) with sizes `a_size` and `b_size`.
2) Pass the `A` vector to `init_vector(...)` with the coresponding seed value `a_seed`, `rand_low`, and `rand_high`.
3) Pass the `B` vector to `init_vector(...)` with the coresponding seed value `b_seed`, `rand_low`, and `rand_high`.
4) Print the elements of the `A` vector by calling the subroutine `print_vector(...)`.
5) Sort the elements of the `A` vector by calling the subroutine `sort_vector(...)`.
6) Print the elements of `A` vector after sorting its elements by calling the subroutine `print_vector(...)`.
7) Print the elements of the `B` vector by calling the subroutine `print_vector(...)`.
8) Search for each value in vector `B` in vector `A` using the linear search algorithm by calling the subroutine `search_vector(...)`.
9) Print the statistical values for the linear search by calling the subroutine `print_stat()`.
10) Search for each value in vector `B` (again) in vector `A` using the binary search algorithm by calling the subroutine `search_vector(...)`.
11) Print the statistical values for the binary search by calling the subroutine `print_stat()`.

Implement the following subroutines:

- `void init_vector(std::vector<int> &vec, int seed, int lo, int hi)`:
  Assign random valued to the elements in `vec` by using the `seed` value. Initialize the random number generator by calling `srand(seed)` and then generate a random number between `lo` and `hi` by using `rand()%(hi-lo+1)+lo`.
- `void print_vector(const std::vector<int> &v, int print_cols, int col_width)`:
  Print the given vector `v` with `print_cols` elements on each line and with each numeric value padded out to `col_width` wide (use std::setw()). See the reference output for the formatting details and alignment. Note that there is an aditional space printed after the element value and before the pipe character `|`.
- `void sort_vector(std::vector<int> &v)`:
  Implement a sort algorithm to sort the elements of vector `v` in ascending order. For this function, use the `std::sort()` function from the STL.
- `int search_vector(const std::vector<int> &v1, const std::vector<int> &v2, bool (*p)(const std::vector<int> &, int))`:
  Implement a generic search algorithm. This will take a pointer to the search routine `p()` that must be called once for each element that is in `v2` to be searched for in `v1`. It must count the number of successful searches and return that value. (Note that this returned value is one of the parameters to be passed to `print_stat()` in your `main()`.
- `void print_stat(int found, int total)`:
  Print the percent of successful searches as a floating-point number on stdout, where `found` is the total number of successful searches and `total` is the size of the test vector that is searched. Note that the reference output includes test printed from `main()` that indicates the type of search and output from `print_stat()` that indicates portion of the output that is the same for both searches and the percentage.
- `bool linear_search(const std::vector<int> &v, int x)`:
  A linear search algorithm, where `x` is the value to search for in vector `v`. It simply starts searching for `x` from the beginning of vector `v` to the end, but it stops searching when there is a match. If the search is successful, it returns `true`; otherwise, it returns `false`. To implement this routine, use the `std::find()` function from the STL: https://en.cppreference.com/w/cpp/algorithm/find Note that `std::find()` requires the use of iterators to specify the range of values to check in `v`. See https://en.cppreference.com/w/cpp/container/vector for a discussion of how to use `vector.begin()` and `vector.end()` to get the iterators needed for `std::find()`. Note that the example on page https://en.cppreference.com/w/cpp/container/vector/begin that shows how to call `std::accumulate()` looks very similar to how you need to call `std::find()`!
- `bool binary_search(const std::vector<int> &v, int x)`:
  A binary search algorithm, where `x` is the value to search for in vector `v`. If the search is successful, it returns `true`; otherwise, it returns `false`. To implement this routine, simply call the `std::binary_search()` function from the STL: https://en.cppreference.com/w/cpp/algorithm/binary_search Note that the example showing how to call `std::binary_search()` is exactly the same way you want to call `std::find()` in your `linear_search()` function!

How The Reference Output Was Created:

- ./twosearch > twosearch.out
- ./twosearch -w3 > twosearch-w3.out
- ./twosearch -l2 -h1002 > twosearch-l2-h1002.out
- ./twosearch -b 18 -c 9 > twosearch-b18-c9.out
- ./twosearch -a250 -b99 -c14 -h1234 -l21 -w5 -x9 -y7 > twosearch-a250-b99-c14-h1234-l21-w5-x9-y7.out
- ./twosearch -x &> twosearch-x.out

Programming Notes:

- Note that the last example reference output run used `&>` to save its output because it fails to run and prints to `cerr` which would not otherwise be saved into the output file for your reference!

Assignment Notes:

- Include any necessary headers and add necessary global constants.
- You are not allowed to use any I/O functions from the C library, such as scanf() or printf(). Instead, use the I/O functions from the C++ library, such as cin or cout.
- Add documentation to the appropriate source files as discussed in your class.

When your program is ready for grading, commit and push your local repository to remote git classroom repository and follow the Assignment Submission Instructions.