CS 135 – Computer Science I
Spring 2016
Programming Assignment 16 (`pa16`)
100 points
Due by 9:00am, 2016-04-23

Problem description

Write a program that measures the *average* time it takes to search and sort an array.

First, write a function that initializes an array of 100,000 integers with random values.

Complete the following steps 3 times, measuring the time to complete each search and sort function:

1. Call array initialization function to initialize an array with random values. After initialization, this array shall remain unaltered for the duration of the program.
2. Call an array copy function to make a working copy of the random array.
3. Call a function that uses the **sequential search** algorithm to locate the value `58980` in the array copy. If found, the function should report its position in the array, or `-1` if not found.
4. Call a function that uses the **bubble sort** algorithm to sort the array copy in ascending order.
5. Call a function that uses the **binary** search algorithm to locate the value `58980` in the array copy. If found, the function should report its position in the array, or `-1` if not found.
6. Call an array copy function to make a duplicate of the random array.
7. Call a function that uses the **selection sort** algorithm to sort the array (described in chapter 8).
8. Call an array copy function to make a duplicate of the random array.
9. Call a function that uses the **insertion sort** algorithm to sort the array.

Your program should calculate and print the *average* CPU time for *each searching and sorting algorithm* and display a neatly formatted table of the results. (See sample output below.)

To find the current CPU time, declare a variable, say `start`, of type `clock_t`. The statement `start = clock();` stores the current CPU time in `start`. You can check the CPU time before and after a particular phase of a program. Then, to find the CPU time for that particular phase of the program, subtract the before time from the after time. Moreover, you must include the header file `ctime` to use the data type `clock_t` and the function `clock`. To convert the difference into seconds, cast to a double and divide by the constant `CLOCKS_PER_SEC`, e.g.

```
static_cast<double>( elapsed ) / CLOCKS_PER_SEC
```

Recommendation: Use a named constant for the array size and set it to a smaller value during development. Don't forget to change and it and test your program using 100,000 elements before submitting!

Sample output:
```
Average CPU times with 5000 elements
Linear search:    0.000009
Binary search:    0.000001
Bubble sort:      0.xxxxxx
Selection sort:   0.xxxxxx
Insertion sort:   0.xxxxxx
```

Submission:

Submit your source code file, named **`pa16.cpp`**, using the `turnin` command.