ECE 1410 Sorting Algorithms Requirements

Description:

Your task is to write 4 C++ class templates that perform the following sorting algorithms on an array of data:

1. Bubble sort (bubble.h)
2. Selection sort (selection.h)
3. Insertion sort (insertion.h)
4. Quick sort (quick.h)

All templates should contain a constructor that does the sorting and a print function that prints the sorted array (elements separated by a single space). All functions take in a pointer to the head of the array and an integer specifying the number of elements in the array. The function prototypes must be as follows:

1. Bubble sort functions:
    a. `Bubble(T *, int);`
    b. `void print(T *, int);`

2. Selection sort functions:
    a. `Selection(T *, int);`
    b. `void print(T *, int);`

3. Insertion sort functions:
    a. `Insertion(T *, int);`
    b. `void print(T *, int);`

4. Quick sort functions:
    a. `Quick(T *, int);`
    b. `void print(T *, int);`

The templates should work on all standard integer and floating-point datatypes (i.e. int, char, float, double, etc.)

Your templates should work with the following main.cpp file:

```
#include <random>
#include <chrono>
#include "bubble.h"
#include "selection.h"
#include "insertion.h"
#include "quick.h"

#define N 100000

using namespace std;
```

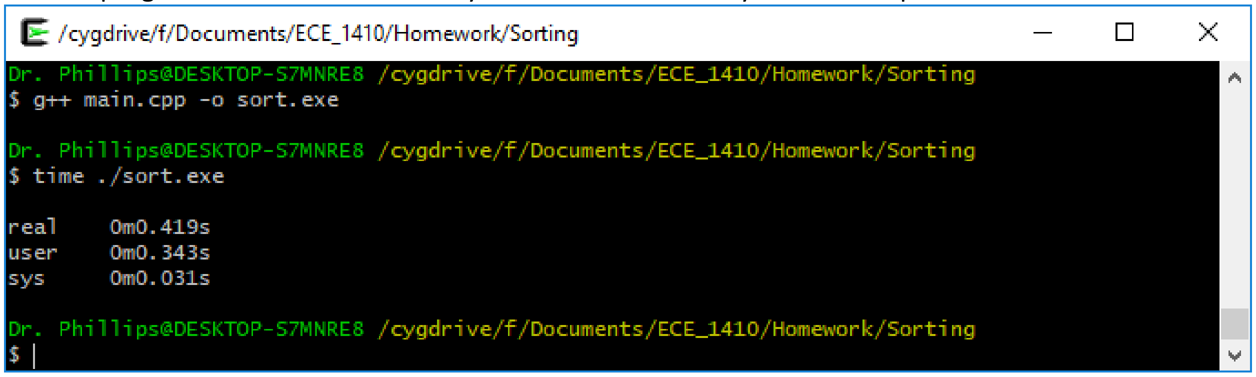
```
int main()
{
  int i, seed, nums[N];

  seed = std::chrono::system_clock::now().time_since_epoch().count();
  std::default_random_engine generator(seed);
  std::uniform_int_distribution<int> distribution(0,999);

  for (i=0; i<N; i++)
  {
    nums[i] = distribution(generator);
  }

  //Bubble <int> s(nums, N);
  //Selection <int> s(nums, N);
  //Insertion <int> s(nums, N);
  Quick <int> s(nums, N);
  s.print(nums, N);

  return 0;
}
```

Notice that one of the 4 sort algorithms should be uncommented for any given compile/run cycle. Also, N can be altered to adjust the size of the array.

Submission:

You should turn in the following to Canvas as a single zip file:

- bubble.h
- selection.h
- insertion.h
- quick.h
- A one-page or more report (as a PDF) analyzing the time performance of the different sorting algorithms and how they relate to Big O notation. Use the Cygwin/Linux time utility to measure program execution time and try different sizes of arrays. Include a plot of the collected data.



```
Dr. Phillips@DESKTOP-S7MNRE8 /cygdrive/f/Documents/ECE_1410/Homework/Sorting
$ g++ main.cpp -o sort.exe

Dr. Phillips@DESKTOP-S7MNRE8 /cygdrive/f/Documents/ECE_1410/Homework/Sorting
$ time ./sort.exe

real    0m0.419s
user    0m0.343s
sys     0m0.031s

Dr. Phillips@DESKTOP-S7MNRE8 /cygdrive/f/Documents/ECE_1410/Homework/Sorting
$ |
```