CISP 430 Data Structures
Project 4: Recursive Sorting

Implement quick sort using one of the following prototype:

Option 1: for a maximum grade of 100%

```
void Quicksort( void* base, size_t nelem, size_t width,
              int (*fcmp)( const void*, const void* ) );
```

Option 2: for a maximum of 75%

```
void Quicksort( ItemType* base, size_t nelem,
     int (*fcmp)( const ItemType&, const ItemType& ) );
```

fcmp accepts two arguments, elem1 and elem2, each a pointer (option 1) or a reference (option 2) to an entry in the table.

The comparison function compares each of the items and returns an integer based on the result of the comparison.

Option 1

```
*elem1  < *elem2        fcmp returns an integer < 0
*elem1 == *elem2        fcmp returns 0
*elem1  > *elem2        fcmp returns an integer > 0
```

Option 2

```
elem1  < elem2        fcmp returns an integer < 0
elem1 == elem2        fcmp returns 0
elem1  > elem2        fcmp returns an integer > 0
```

In the comparison, the less-than symbol (<) means the left element should appear before the right element in the final, sorted sequence. Similarly, the greater-than (>) symbol means the left element should appear after the right element in the final, sorted sequence.

NOTE: this is the same method used by the built-in function strcmp

base: points to the first element of the array
nelem: the number of elements in the array
width: the number of bytes in each element (option 1)

You will compare this sorting method to the built-in Quick-sort, qsort. Run each method 100 times using 10000 randomly generated numbers and compare the time that each takes. Use the built-in time function, **`ftime`**. With this function, you should be able to time to the nearest millisecond. Do not include the time that it takes to fill your arrays with random number; time the sorts only.

You will have three files:

1. Quicksort1.h or Quicksort2.h
2. Quicksort1.cpp or Quicksort2.cpp
3. Main.cpp

Quicksort#.h will contain the prototype for quick sort and nothing else
Quicksort#.cpp will contain the implementation for quick sort and nothing else
Main.cpp will contain function main and fcmp

Here '#' represents the option you have selected. Make sure that you have '1' if you are submitting option #1 and '2' if you are submitting option #2

When you implement your version of quick sort, you must use the method that we discuss in class. If you use some other method you will receive **ZERO POINTS** for this assignment.