```
/*
 * Shiwei Huang
 * CS345: huangP1.cpp
 * Purpose: Usings threads to sort a list of unsorted numbers from a .txt file.
            Break the original unsorted list from .txt file into two threads.
            Feed the parts to separate threads (2 threads in my case) to do the sorting.
            Output the sorted list into sorted1.txt and sorted2.txt
            Make the last file and make the final sorted list, called finalAnswer.txt
 *
 * Input: a .txtfile contains data, which is a list of integers to be sorted.
   The first line of the file will contain N, which is number of values in the file.
   This is followed by N lines, where each line contains one integer value.

 * Output: The program will create three text files (in my case).
   The first two file (sorted1.txt, sorted2.txt) is sorted sublists created by each thread.
   The last file is the final sorted list, called finalAnswer.txt
 *
 * Assignment purpose: threads, bubble sort, merge sort, make files, write files, struct, pointers
     etc.
 *
 * Specifics: The main function will compute the final sorted list by writing the finalAnswer.txt
     file.
              This should be done in linear time O(N).
              The sorting function that used to sort each sublist could be any sorting function.
              Eg: insertion sort or bubble sort.
 */

#include <iostream>
#include <string>
#include <fstream>
#include <pthread.h>
#include <unistd.h>

using namespace std;

struct referenceArray {
    int half; // which half of the thread is computing;
    int totalNumber; // total number of integers N from the .txt input file;
    int *arrayPointer; // from the referenceArray struct, making a pointer pointing the array which
        is going to be constructed under the main;
};


void * bubble_sort ( void * ); // in the thread, using bubble sort function to sort two sublists;

int main() {

    ifstream file("number.txt"); // Copyright: http://cs.wheatonma.edu/mgousie/comp345/randomNums.
        txt
    if( file.is_open() ) {

        int n;  //variable 'n': n = total number of integers from the number.txt file; which is
            also the first line from the file;
        int nextInt;
        file >> nextInt; //reads the first number from the file;
        n = nextInt; // store the first line from the file (total number of integers) into variable
            n;

        int *myArray = new int[n]; //making a dynamic array containing n integers;

// writing numbers into the array
        for( int i = 0; i < n; ++i ) { //using for loop to put the numbers from the .txt file to
            myArray;
            file >> nextInt; //Also note that nextInt is starting from the second number;
            myArray[i] = nextInt;
        }
```

```
        pthread_t thread1, thread2; //2 threads;
        int half1, half2; //identify thread number;

        struct referenceArray * ptr1; //create two structs, one for each thread;
        struct referenceArray * ptr2;


        ptr1 = new referenceArray; // Pointer pointing to my new struct location;
        ptr2 = new referenceArray;

        ptr1 -> totalNumber = n; // store n into totalNumber, putting in my struct;
        ptr2 -> totalNumber = n;

        if ( n % 2 == 0 ) { // total number N from the .txt file is even;
            ptr1 -> arrayPointer = &myArray[0];
            ptr2 -> arrayPointer = &myArray[n/2];
        }

        else { // total number N from the .txt file is odd;
            ptr1 -> arrayPointer = &myArray[0];
            ptr2 -> arrayPointer = &myArray[n/2+1];

        }


        ptr1 -> half = 1;
        pthread_create ( &thread1, NULL, bubble_sort, (void *) ptr1);

        ptr2 -> half = 2;
        pthread_create ( &thread2, NULL, bubble_sort, (void *) ptr2 );

        // must wait for children to finish before making further steps; which is to sort two
            sublists to the finalAnswer.txt file;
        pthread_join (thread1, NULL); // wait for thread1 to finish;
        pthread_join (thread2, NULL); // wait for thread2 to finish;



// making the final sorted list by combining all of the sorted sublists;
// This step should be done in linear time O(N);

        if ( n % 2 == 0 ) { //even number of integers (N) from the .txt file.
            int i = 0; //location counter for the first part of the array;
            int j = 0; //location counter for the second part of the array;
            ofstream myFile;
            myFile.open ("finalAnswer.txt");

            // making number comparison from two parts of the array
            if ( myArray[i] < myArray[n/2+j] ) { // if the first number from the first part of the
                array is less than the first number from the second part of the array;
                myFile << myArray[i] << endl; //write down this number to the file;
                i++; //move my counter in the first part of my array;
            }
            else {
                myFile << myArray[n/2+j] << endl; //else, move my counter in the second part of the
                    array;
                j++;
            }
            //while the counter have not reaching to the end of ( first part of the array ) &&
                ( second part of the array )
            while ( ( i < n/2 ) && ( j < n/2 ) ) {
                if ( myArray[i] < myArray[n/2+j] ) { // if the number from the 1st parts of my
                    array is less than the number from the 2nd parts of my array;
                    myFile << myArray[i] << endl; //write down the small number;
                    i++;
                }
                else {
                    myFile << myArray[n/2+j] << endl;
```

```
                j++;
            }

        }//while;

        // write down the remaining number from the array (if there are any number left)
        if ( i > j) {//if first part of the array is already printed (because i > j); then I
            need to finish printing from the second part of the array;
            for ( int k = j; k < n/2; k++ ) {//k is the already printed part of the second part
                of the array;
                myFile << myArray[i+k] << endl; //myArray[i+k] is the remaining parts of the
                    second part of the array;
            }
        }
        else {//if second part of the array is already printed; then I need to finish printing
            from the first part of the array;
            for ( int k = i; k < n/2; k++ ) {
                myFile << myArray[k] << endl; //myArray[k] is the remaining parts of the first
                    part of the array;
            }
        }
        myFile.close();

    }//if. even number of integers (N) from the .txt file;

    else {//odd number of integers (N) from the file;
        int i = 0;
        int j = 0;

        ofstream myFile;
        myFile.open ("finalAnswer.txt");

        if ( myArray[i] < myArray[n/2+1+j] ) { // if the first number from the first part of
            the array is less than the first number from the second part of the array;
            myFile << myArray[i] << endl;
            i++; //move my counter in the first part of the array;
        }
        else {
            myFile << myArray[n/2+1+j] << endl; //else, move my counter in the second part of
                the array;
            j++;
        }
        //while the counter have not reaching to the end of ( first part of the array ) &&
            ( second part of the array )
        while ( ( i < n/2+1 ) && ( j < n/2 ) ) {
            if ( myArray[i] < myArray[n/2+1+j] ) { // if the number from the 1st parts of my
                array is less than the number from the 2nd parts of my array;
                myFile << myArray[i] << endl;
                i++;
            }
            else {
                myFile << myArray[n/2+1+j] << endl;
                j++;
            }

        }//while;

        if ( i > j) {//if already finished printing first part of the array; then I am output
            the remaining part of the second part of the array;
            for ( int k = j; k < n/2; k++ ) {
                myFile << myArray[i+k] << endl;
            }
        }
        else {//if already finished printing second part of the array;
            for ( int k = i; k < n/2+1; k++ ) {
                myFile << myArray[k] << endl; //then I am outputing first part of my array;
            }
        }
```

```
            myFile.close();

        }//else. odd number of integers (N) from the file;


    }//if
    else {
        cout << "Error: File cannot found" << endl;
    }//else

    return 0;
} //main;

// separate threads;
// bubble_sort take an pointer and pass by reference;
void * bubble_sort (void * ptr) { //passing the pointer and make it pass by reference;

    struct referenceArray * local_ptr = ( struct referenceArray *) ptr; // making void pointer and
        recast it back to struct referenceArray;
    int half = local_ptr -> half;
    int *arrayPointer = local_ptr -> arrayPointer;
    int totalNumber = local_ptr -> totalNumber;

    if ( half == 1 ) { //thread1;
        if ( totalNumber % 2 == 0 ) {//totalNumber N from the file is even;
            // Bubble sort function
            for ( int iteration = 1; iteration < totalNumber/2; iteration++ ) {
                for ( int index = 0; index < totalNumber/2 - iteration; index++ ) {
                    if ( local_ptr -> arrayPointer[index] > arrayPointer[index+1] ) {
                        int temp = local_ptr -> arrayPointer[index];
                        local_ptr -> arrayPointer[index] = arrayPointer[index+1];
                        local_ptr -> arrayPointer[index+1] = temp;
                    }//if
                }//for
            }//for

     // write down sorted number into the file
            ofstream myFile;
            myFile.open ("sorted1.txt");
            for ( int i = 0; i < totalNumber/2; ++i ) {
                myFile << local_ptr -> arrayPointer[i] << endl;
            }
            myFile.close();


        } else { //totalNumber N from the file is odd;
            for ( int iteration = 1; iteration < totalNumber/2+1; iteration++ ) {
                for ( int index = 0; index < totalNumber/2+1 - iteration; index++ ) {
                    if ( local_ptr -> arrayPointer[index] > arrayPointer[index+1] ) {
                        int temp = local_ptr -> arrayPointer[index];
                        local_ptr -> arrayPointer[index] = arrayPointer[index+1];
                        local_ptr -> arrayPointer[index+1] = temp;
                    }//if
                }//for
            }//for


            ofstream myFile;
            myFile.open ("sorted1.txt");
            for ( int i = 0; i < totalNumber/2+1; ++i ) {
                myFile << local_ptr -> arrayPointer[i] << endl;
            }
            myFile.close();
        }//else. totalNumber in file is odd;

    } //thread1

     else { //thread2;
```

```
            for ( int iteration = 1; iteration < totalNumber/2; iteration++ ) {
                for ( int index = 0; index < totalNumber/2 - iteration; index++ ) {
                    if ( local_ptr -> arrayPointer[index] > arrayPointer[index+1] ) {
                        int temp = local_ptr -> arrayPointer[index];
                        local_ptr -> arrayPointer[index] = arrayPointer[index+1];
                        local_ptr -> arrayPointer[index+1] = temp;
                    }//if
                }//for
            }//for


            ofstream myFile;
            myFile.open ("sorted2.txt");
            for ( int i = 0; i < totalNumber/2; ++i ) {
                myFile << local_ptr -> arrayPointer[i] << endl;
            }
            myFile.close();



        }//else; finish thread2;


    }//void
```