Sorting Contest - Prog 3
CECS 325-02 System Programming with C++
Spring 2024
Due: Tuesday March 12, 2024

Linux has a sort command. If you wanted to use the linux sort command to sort all the items in a file called "numbers.dat", you could use it like this:

```
$ sort numbers.dat        ←This command treats the contents of the file like text
$ sort -n numbers.dat     ← this command treats the contents of the file like numbers
$ sort -c numbers.dat     ← this checks to see if the file is sorted
```

You could also sort the file like this:

```
$ cat numbers.dat | sort      ←send output to stdout using cat then pipe that to the sort command
$ more numbers.dat | sort     ←send output to stdout using more then pipe that to the sort command
```

You could sort the first or last 1000 numbers like this:

```
$ head -1000 numbers.dat | sort     ← sort the first 1000 lines, send output to stdout
$ tail -1000 numbers.dat | sort           ← sort the last 1000 lines
```

Each of the above commands will display the sorted numbers (lines) on the screen. Instead of displaying the numbers, you could write them to a file called "sorted.out"

```
$ cat numbers.dat | sort > sorted.out  ← redirect output to a file called "sorted.out"
$ sort -n numbers.dat > sorted.out
```

In this assignment you will generate a file that has 1 million numbers between -100000 and +100000. Write these numbers to a file called "numbers.dat". Then run this Linux command to see how long it takes to sort 1 million integers:

```
$ time sort -n numbers.dat  > sorted.out
```

Here's how fast it took on my computer:

```
steve@SGOLD-NB2021:~/prog3$ generate 1000000 -100000 100000
argv[0]:generate
argv[1]:1000000
argv[2]:-100000
argv[3]:100000
steve@SGOLD-NB2021:~/prog3$ time sort -n numbers.dat > sorted.out

real    0m1.145s
user    0m3.045s
sys     0m0.233s
steve@SGOLD-NB2021:~/prog3$
```

Your job is to write a sort program that uses the bubble sort.

This will take some time to do and we don't want to wait around for the results, so we will run the program in the background using the &.

You will run this shell file to test your program:

```
# this file is called sortrace.sh
# it must have execute privilege set to run
# run it as a background task like this:
#               $ rm sortrace.log                    # start with fresh log file
#               $ sortrace.sh >> sortrace.log &      # this may take an hour
echo Generating 1000000 random numbers
sleep 1
generate 1000000 -100000 100000          # you have to write generate.cpp
sleep 1
echo Starting system sort
sleep 1
{ time sort -n numbers.dat > systemsort.out; } 2>> sortrace.log
sleep 1
echo Starting mysort
sleep 1
{ time mysort numbers.dat mysort.out; } 2>> sortrace.log    # you have to write mysort.cpp
sleep 1
sort -c -n mysort.out 2>> sortrace.log                  # verify file is sorted
```

You must write the following programs

- generate.cpp
  - **Purpose:** generate a file called numbers.dat
  - **Description:** The numbers.dat file will have COUNT random numbers between MIN and MAX
  - **Usage**: generate COUNT MIN MAX
  - **Example**: generate 1000000 -100000 100000
  - This program accepts command 3 command line arguments as shown above
  - If there are not 3 command line arguments, the program fails and prints error message
- mysort.cpp
  - **Purpose**: read numbers from input file, sort, write sorted numbers to output file.
  - **Description**: Uses bubble sort function on an array of integers
  - It will accept up to 1 million numbers from the input file, but will run successfully with less
  - It accepts 2 command line arguments which is the input file and the output file name

Requirements:

Use an array to store your numbers – do not use a vector

You must have a function called bubble(int A[ ], int size)

The bubble function takes 2 parameters – the array and the number of items to sort.

The bubble function will be called from the main function
The other generate and mysort programs must work according to the above descriptions

What to submit:

- mysort.cpp
- generate.cpp
- sortrace.log
- sortrace.sh

Notice – Your requirement is to use the exact sortrace.sh file listed above which sorts 1,000,000 numbers. Mac users may need to make minor changes. However the graders will run your program with 10,000 numbers for efficiency so make sure it works with less than 1,000,000 numbers as described in the requirements.