Coding Assignment 1

CSE2320

Name your program `Code1_xxxxxxxxxx.c` where xxxxxxxxx is your student id (not netid). My file would be named `Code1_1000074079.c`.

Please place the following files in a zip file with a name of `Code1_xxxxxxxxxx.zip` and submit the zip file.

`Code1_xxxxxxxxxx.c`

`Output_xxxxxxxxxx.xls`

`TestFile.txt`

A zip file is used to avoid Canvas's file renaming convention.

Reminder – ANY compiler warning(s) or error(s) when compiled on Omega will result in an automatic 0. This apply no matter what the warning/error is. You will need to code so that your final program will compile cleanly and run on both the VM and Omega.

For example, if the GTA compiles your code on Omega and sees

```
[frenchdm@omega CA1]$ gcc Code1_1000074079.c
Code1_1000074079.c:44:2: warning: no newline at end of file
```

you will receive a 0 on the assignment.

Step 1

Create a program than can run Merge Sort on a hardcoded small array. You can copy the code from the lecture slides or copy it from an Internet source. The code MUST BE in C. If you are copying from an Internet source, make sure you are getting MERGE SORT and not some other version of sort. Your assignment will not be graded and you will be assigned a grade of 0 if the code you submit is not in C and is not merge sort.

NOTE : At this point, you can take a copy of this working code and add some print statements and practice for OLQ3. Change the hardcoded array and see if your printouts match the values you manually came up with – start with the array explained in lecture and see if your program and your manual calculations match the OLQ3 Review and Practice document.

Add the ability for your program to also run Insertion Sort on a hardcoded small array. You can copy the code from the lecture slides or copy it from an Internet source. The code MUST BE in C. If you are copying from an Internet source, make sure you are getting INSERTION SORT and not some other version of sort. Your assignment will not be graded and you will be assigned a grade of 0 if the code you submit is not in C and is not insertion sort.

The merge sort code will run and then the insertion sort code will run. They are both in the same program to get both process approximately the same run environment (rather than put them in separate programs).

Step 2

Download the file, FileGenerator.e, from Canvas to either your VM or Omega.

The executable should run on your VM with no issues. If you put it on Omega and get a “Permission denied” error when you try to run it, then you will need to run the following command to change it to an executable so that it will run on Omega.

```
[frenchdm@omega CSE2320]$ FileGenerator.e
-bash: ./FileGenerator.e: Permission denied
[frenchdm@omega CSE2320]$ chmod +x FileGenerator.e
```

Run it and it will give you a message about what parameters it expects. Use this executable to generate the following 7 files. You can name them what you want but you need to create each file containing the required number of records per file. Either create them on Omega or on your VM and then copy them to the other system. Do NOT create them on both systems – you want to use the exact same files on both systems.

1,024
10,000
50,000
100,000
500,000
1,000,000
2,000,000

Run the file generator one more time and create a file of 10 records. Use this file for testing and submit in your zip as "`TestFile.txt`".

Step 3

Add the ability to accept command line parameters. The "Review Materials" module contains the CSE 1320 lecture on command line parameters. The file name that the program will read will be provided on the command line.

Step 4

Add the ability to read a file name from the command line. You will run your program as

Executable_name LargeFile.txt

or

Executable_name VeryLargeFile.txt

Open the file (you should have gotten the file's name from `argv[1]`) read only ("r") since we are only reading and not writing to the file. Using `fgets()`, loop through the file and count the number of lines in the file. Use `fseek()` to move the file pointer back to the beginning of the file (See "File Handling in C" in the "Review Materials" module of Canvas).

Step 5

Use `malloc()` with the number of lines you counted in the file to dynamically create an `int` array on the heap. Now read through the file again and put each line (each line is a number) in an array element. This process will allow you to create arrays of various sizes.

Run Merge Sort on this array and then `free()` it and `malloc()` it again and fill it again and then run Insertion Sort. Do not run both sorts on the same copy of the array. Do not try to make a copy of the array – this assignment uses VERY large arrays and you will not have enough memory to hold 2 copies. So `malloc()`, read file into array, call merge sort, `free()` array, `malloc()` array again, read file into array, call insertion sort, `free()` array.

Step 6

Add a function to print the array. Here is a suggested version – you are not required to use this exact code but your version must output the same – one number per line.

```
void PrintArray(int ArrayToPrint[], int SizeAP)
{
      int i;

      for (i = 0; i < SizeAP; i++)
            printf("%d\n", ArrayToPrint[i]);
```

```
}
```

Add conditional compile statements around the call to this function (which should be called before and after sorting the array).

```
#ifdef PRINTARRAY
PrintArray(A, elements);
#endif
```

Using `TestFile.txt`, run your new version and confirm that it sorts correctly and uses the file as input by compiling your code with the conditional compile.

```
gcc Code1_xxxxxxxxxx.c -D PRINTARRAY
```

It will be harder and harder to see the proper sorting as your file grows so confirm your coding with smaller tests. The conditional compile will allow the GTA to test your program as well. Compile without the conditional compile when you start running the tests so that you do not print the larger arrays.

Step 7

Add the code to time your sort. To do this, define two variables of type `clock_t`

```
clock_t start, end;
```

You then call the `clock()` function to get the start time. You will call the `clock()` function again after the merge to get the end time.

```
start = clock();
```

Call Merge Sort

```
end = clock();
```

print the total tics by subtracting `start` from `end`

You will need to repeat this process to time the Insertion Sort.

Be sure to only include the actual sort code in the time – do not include reading the file or setting up the array. We are just going to use the actual number of clock tics so do not divide your total ticks by anything. Print out the tics needed by each sort (see video for example).

Step 8

Download "Output_xxxxxxxxxx.xls" from Canvas. Change the x's to your student id and save the spreadsheet. You will be submitting this spreadsheet with your assignment. Watch the video "How to Chart" to see how the chart of the Big O values that is already in the spreadsheet was created. You will need to be able to repeat this process with the Omega and VM data you will be gathering.

Step 9

Run your 7 files on your VM and on Omega. Record the number of tics output by the program for each file of *n* records. Once you have recorded all of those values in your spreadsheet, create a chart of the "n and VM tics" and a chart of the "n and Omega tics" using the same technique used with the Big O values in the video. All 3 graphs will be in the "Results" tab of the spreadsheet.

NOTE : it is highly likely that Omega will timeout and shut down your terminal before your larger files can finish running on Omega. If this happens, then you need to output your results to a file and run the process in the background. See video "Run background process" to see an example of how to do this.

Step 10

Write a paragraph (at least 3 sentences) about your conclusions. What do the 3 charts show you? How similar did your graphs for each sort come out on the VM vs Omega vs Big O Notation? Were you surprised by the differences between the run times of Merge Sort vs Insertion Sort? How did your tics on Omega compare to the VM? Write your paragraph in the box in the "Conclusions" tab of the spreadsheet.