

CS 109 – C/C ++ Programming for Engineers w. MatLab– Spring 2012

Homework Assignment 3 – First Draft

Generating an Image from Sonar Data Files

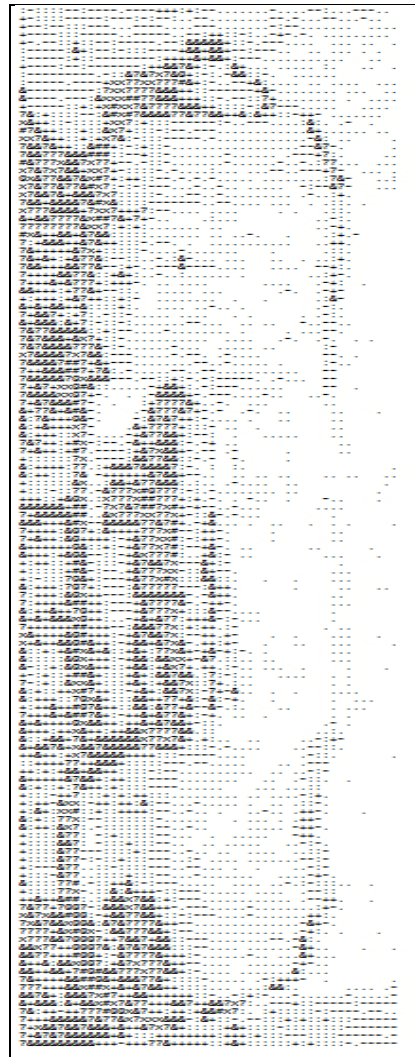
Due: Wednesday February 15th by 8:00 a.m., via Blackboard. Optional hard copy may be turned in during lab.

Overall Assignment

For this assignment you are to write a program that reads in sidescan sonar data from an input file specified by the user, and generates a character-based graphical image of the data, to be displayed on the screen and also stored into an output data file specified by the user.

Background: Sidescan Sonar Images

Shown below is a sidescan sonar image of the Straits of Mackinac, a shipwreck located in southern Lake Michigan northeast of Chicago, and an ASCII representation of the corresponding sonar data. Your task will be to write a program that reads in (pseudo) sonar data and generates images similar to those shown on the right below.



Background: Data File Format

The input data files that your program needs to read begins with a line containing a single integer, representing the number of data samples collected for each ping of the sidescan sonar. This tells you how many data values are on each line of the file following this initial line.

The remainder of the file contains floating point numbers in the range 0.0 to 1.0 inclusive. Each line corresponds to one ping of the sonar, and contains as many data items as the integer read in on the first line of the file.

For this assignment we will assume that the data samples are taken on regular distance intervals away from the boat, and that the boat moves by the same distance between consecutive pings. (In practice neither of these assumptions would be valid. Echo samples are collected at even *time* intervals, and the software would need to calculate the distance away from the boat as you did in homework assignment 2, assuming a flat bottom. The movement of the boat would be determined from GPS data collected in a separate file, and the boat may also turn, further complicating matters.)

So the beginning of a data file might look something like this:

```
50
3.6328125e-001  2.0703125e-001  3.4375000e-001  3.3593750e-001  3.4765625e-001
4.1796875e-001  2.9687500e-001  3.2421875e-001  3.3593750e-001  3.0078125e-001
4.0234375e-001  2.1484375e-001  2.8125000e-001  3.7500000e-001  2.4609375e-001
4.0625000e-001  2.5000000e-001  2.0703125e-001  2.5390625e-001  2.4609375e-001
4.0234375e-001  2.1484375e-001  1.9531250e-001  2.6953125e-001  2.6562500e-001
3.6328125e-001  2.4609375e-001  2.7734375e-001  2.8125000e-001  2.7734375e-001
3.6718750e-001  2.6171875e-001  2.6171875e-001  2.8125000e-001  2.9687500e-001
```

Program Details

For this assignment you are to write a computer program that reads in data from a data file as described above, and then prints ASCII characters to the screen and into an output file corresponding to the data.

- Your program should first print out your name and ACCC account name (e.g. astudent), and explain to the user what the program does.
- Your program should then ask the user for the names of input and output data files, and verify that they can be opened properly. (Each file should be checked individually. If a file cannot be opened, ask the user for a different file name.)
- Next your program reads in the number of data samples on each line from the file
- Next your program enters a loop to read the rest of the file, up until the end of the file is reached. For each line of the file your program should use a nested loop to read each of the data values on the line, and print out a single character depending on the data value, as described below under output. The characters should be printed to both the screen and the output data file, and each line should be terminated by a newline character.
- After the end of the file is reached, your program needs to close the input and output files, and report the size of the image (width x height) read from the file.

Output

The characters to be printed for each input data value are as follows:

Data value	Character
Less than 0.1	space
0.1 to 0.2	.
0.2 to 0.3	-
0.3 to 0.4	:
0.4 to 0.5	+
0.5 to 0.6	&
0.6 to 0.7	?
0.7 to 0.8	X
0.8 to 0.9	#
greater than 0.9	@

So the beginning output of the sample data file might look something like:

```
: - : : : - - : - - - . - - - + + + : + : - . . . . . - . . . . - : . . . - - . .
+ - : : : : - - - - : - - - : - - - : . . - . . . . . - - . - . . . - . . . - .
+ - - : - - : : : - - - . . - - - . . . - - . . . . . - : - - . . . . . . . . .
+ - - - : : : - - - . . - - - . . - : + + : - : . . + - . - . . . . . . . . .
+ - . - - : + : : - - : - - - . - - : & & & & + : - . - - . . . . . . . . . .
: - - - - : & + : - - - - : : : - - - + & & + & & + - - - . . . . . . . . . .
: - - - - : + : : - - - - - : + + + + & + & + : - - . . . . . . . . . . . .
+ - - - - : - - : - - - - - : + + & ? & + : - . : & + . . . . . : . . . . .
: - - - - - . - : : & ? & ? X ? & @ + : - : . - & & : - . . . . . . . . . .
: - - - - . - - - + X X ? ? X X ? ? ? # & + : - . - - + & : . . . . . . . . . .
& - - - - - : ? X X ? ? ? ? & & + + : - - - - + & - . . . . . . . . . . . .
& - - - - . - - - : & X X X # # ? ? & & + + + : - . - - : ? + . . . . . . . . . .
+ - - - - : : + : + X # X X ? & ? ? ? ? & & + + : : : - : & ? - - . . . . . . . . .
? & : + : : : - - : & # X # ? & & & ? ? & ? ? & & + + & : & + + : - + + - . . . . . . . . .
```

Special Notes:

- If your program detects problems in the data file, such as a non-positive number for the number of data samples, not enough data, or numbers outside the range of 0.0 to 1.0, then you should detect and report this. It is acceptable to end the program once problems are detected.

What to Hand In:

1. Your code, **including a user documentation file**, should be handed in electronically using Blackboard.
2. The intended audience for the documentation file is a general end user, who might want to use this program to perform some work. They do not get to see the inner workings of the code, and have not read the homework assignment. You can assume, however, that they are familiar with the problem domain (e.g. sidescan sonar.)
3. A secondary purpose of the documentation file is to make it as easy as possible for the grader to understand your program. If there is anything special the grader should know about your program, be sure to document it in the documentation file. In particular, if you do any of the optional enhancements, then you need to document what they are and anything special the TA needs to do to run your program and understand the results.
4. If there are problems that you know your program cannot handle, it is best to document them as well, rather than have the TA wonder what is wrong with your program.
5. Make sure that your name appears at the beginning of each of your files. Your program should also print this information when it runs.

Optional Enhancements:

It is course policy that students may go above and beyond what is called for in the base assignment if they wish. These optional enhancements will not raise any student's score above 100 for any given assignment, but they may make up for points lost due to other reasons.

- Other enhancements that you think of – Check with TA for acceptability.