CSCI-21 Lab 10, due 12/8/22

Please work in groups of two for this. Do not work alone. This will lead directly into the final programming project, which will use interrupts instead of polling.

You may turn off branch and load delays for this program. It will make it somewhat simpler. Be sure to turn on memory-mapped I/O.

Base this on the example program memmapIO.asm. Make sure you fully understand what memmapIO.asm is doing before you start programming.

You may not use `syscall` for input, because the program would then block on the input. Instead, use a polling routine to catch input (and output). **You must do ALL input and output through the memory-mapped input/output ports, NOT through `syscall`.**

Write a program that starts with two arrays of characters (an `.asciiz` string of at least 60 characters, labeled '`source`', and a `.space` buffer of greater length, labeled '`display`'. Initially, `source` must contain a character string with a '`\n`' before the terminating NUL. Include upper- and lower-case letters, digits and punctuation characters. Start your program by copying the `source` string into the `display` buffer. Use a subroutine (using the "real-world" calling convention) to do this, passing in the bases of the two blocks of memory as parameters on the stack.

Then loop forever, alternately polling the input and output ports to see if the devices are ready to accept a character (output) or give you a character (input).

If the output device is ready, display the next character in the display string. When you display the '`\n`', start over at the beginning of the display buffer.

If the input device has a character ready, get it and process it according to these rules:

'`s`': sort the `display` array using any simple sort routine (bubble or ripple is fine). Do not sort the '\n' in with the rest of the characters.

'`t`': toggle the case of every alphabetic character in the `display` array (for example, 'T' becomes 't', 't' becomes 'T' and all non-alphabetic characters stay unchanged).

'`a`': replace the `display` array elements with the `source` elements once again.

'`r`': reverse the elements in the `display` array. Do not reverse the '\n' to the front!

'`q`': terminate program execution gracefully, using syscall 10.

Ignore any other input characters.

Note: the line being displayed could change mid-line, since input can come in at any point. You will might want to code delay loops in your program to keep the output lines from flying down the screen too quickly (just loop to a few hundred or thousand in a loop with an empty body).