CSCI-21 Lab 8, due 5/16/22

You may work in groups of two or three for this. This will lead directly into the final programming project, which will use interrupts instead of polling. Your final project must be your own work.

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.

You should 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 input routine catch input. Nor may you use syscall for output. **You must do ALL input and output through the memory-mapped input/output ports, NOT through syscall.**

Write a program that starts with a string of characters (an .asciiz strings), of at least 40 characters, labeled 'source'; and a .space directive labelled display larger than the source string. Initially, the source array contains a character string with a '\n' before the terminating NUL. Start your program by copying the source array into the display area. Use a little subroutine to do this, passing in the bases of the two blocks of memory.

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). The structure of this loop is like this:

```
loop:
   if( keyboard ready)
      extract and process a character
   end if
   if(display ready)
      display a character
   end if
end loop
```

You may use subroutines to extract and process the input character or to display the output character to the console.

If the output device is ready, display the next **single** character in the display string, then save the position for the next time the display is ready. When you have printed 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 easy sort routine (bubble or ripple is fine). Do not sort the '\n' with the rest of the characters.

't': toggle the case of every alphabetic character (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 might want to code delay loops in your program to keep the output lines from flying down the screen.