Problem 1 - A simple loop in MIPS

Recall that the factorial, \( n! \), of \( n \) is given as follows:

\[
0! = 1 \\
n! = n \cdot (n - 1)! \quad n > 0
\]

Write a MIPS program which takes a non-negative integer \( n \) in \$1\) and stores \( n! \) in \$3\).

Problem 2 - More loops in MIPS

Recall that the Fibonacci sequence can be defined as follows:

\[
f_0 = 0 \\
f_1 = 1 \\
f_{n+2} = f_{n+1} + f_n 
\quad n \geq 0
\]

Write a MIPS program which takes a non-negative integer \( n \) in \$1\) and stores \( f_n \) in \$3\).

Problem 3 - Arrays in MIPS

Thus far we’ve only written programs which accept two integers as arguments, but with mips.array we can also write programs which manipulate arrays. Write a MIPS program which accepts the address of an array in \$1\) and its length in \$2\) and stores the product of the numbers in the array in \$3\).
Problem 4 - Basic I/O in MIPS

Recall that you can read from stdin and write to stdout by loading from or storing to addresses \texttt{0xffff0004} and \texttt{0xffff000c} respectively. Note that EOF is represented by \texttt{-1}, and otherwise a single byte will be read or written at a time.

Write a MIPS program which reads in two characters from stdin (you may assume EOF is not encountered) and prints out the character \texttt{1} if the first is less than the second, or \texttt{0} otherwise. It should then print a newline.

Problem 5 - I/O and loops in MIPS

Adapt your solution to problem 4 to read in characters from stdin until EOF is encountered and print out their uppercase versions to stdout. If the character is not a lower-case letter, simply print out the character unchanged.

Problem 6 - Using the stack in MIPS

Write a MIPS program which reads in characters from stdin until EOF is encountered, then prints the same characters out backwards to stdout. Use the stack to store the characters.

Problem 7 - Functions and recursion in MIPS

Rewrite your solution to Problem 1 using a recursive function instead of a loop.