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 $\$3$.

Solution:

```mips
; Initialize the answer ($\$3$) = 1 and $\$11$ = 1
lis $\$3
.word 1
add $\$11, $\$3, $0

; Loop until $\$1 = 0
loop: beq $\$1, $0, end

; $\$3 = $\$3 \cdot $\$1
mult $\$3, $\$1
mflo $\$3

; Go to next index ($\$1 = $\$1 - 1)
sub $\$1, $\$1, $\$11
beq $0, $0, loop

end: jr $31
```
Problem 2 - More loops in MIPS

Recall that the Fibonacci sequence can be defined as follows:

\[
\begin{align*}
    f_0 &= 0 \\
    f_1 &= 1 \\
    f_{n+2} &= f_{n+1} + f_n & n \geq 0
\end{align*}
\]

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

Solution:

\[
\begin{align*}
; &\$3 = f_i, \$4 = f_{i+1} \\
; &\$11 = 1 \\
add &\$3, \$0, \$0 \\
\text{lis} &\$4 \text{ word 1} \\
add &\$11, \$4, \$0
\end{align*}
\]

;Loop until \$1 = 0
loop: \text{beq} \$1, \$0, end

\[
\begin{align*}
; &\$5 = f_{i+1} \\
\text{add} &\$5, \$4, \$0 \\
; &\$4 = f_{i+2} = f_i + f_{i+1} \\
\text{add} &\$4, \$3, \$4 \\
; &\$3 = f_{i+1} \\
\text{add} &\$3, \$5, \$0
\end{align*}
\]

;Go to the next iteration (\$1 = \$1 - 1)
\text{sub} \$1, \$1, \$11 \\
\text{beq} \$0, \$0, loop

end: \text{jr} \$31
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.

Solution:

```
; $2 = 4 * $2 + $1
add $2, $2, $2
add $2, $2, $2
add $2, $2, $1

lis $4
.word 4

lis $3
.word 1

; Loop until $1 = $2, incrementing $1 by 4 every time
loop: beq $1, $2, end

; $5 = *$1 = A[i]
lw $5, 0($1)

; $3 = $3 * $5
mult $3, $5
mflo $3

; Go to next index
add $1, $1, $4
beq $0, $0, loop

end: jr $31
```
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 0xffff0004 and 0xffff000c respectively. Note that EOF is represented by −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 1 if the first is less than the second, or 0 otherwise. It should then print a newline.

Solution:

```
; $27 is stdin, $28 is stdout
lis $27
.word 0xffff0004
lis $28
.word 0xffff000c

; $20 is the ’0’ character
lis $20
.word 48 ;0x30 in hex

; Load characters from stdin
lw $3, 0($27)
lw $4, 0($27)

; $3 = 1 if $3 < $4, 0 otherwise
slt $3, $3, $4

; Note that ’0’ + 0 = ’0’ and ’0’ + 1 = ’1’
add $20, $20, $3

; Print the character to stdout
sw $20, 0($28)

; Newline is 10 = 0xA, so load and print it
lis $20
.word 10
sw $20, 0($28)

jr $31
```
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.

Solution:

; $27 is stdin, $28 is stdout
lis $27
.word 0xffff0004
lis $28
.word 0xffff000c

; $25 = 'a', the first lowercase letter.
; Characters less than $25 are not lowercase.
lis $25
.word 97

; $26 = 'z', the last lowercase letter.
; Characters more than $26 are not lowercase.
lis $26
.word 122

; $20 is 'A' - 'a', the amount that
; we should add to make a character uppercase
lis $20
.word -32

; $24 is EOF
lis $24
.word -1

; Load characters until EOF is encountered
loop: lw $3, 0($27)
beq $3, $24, end

; If $3 < $25, we can print this unchanged
slt $5, $3, $25
bne $5, $0, print

; If $26 < $3, we can print this unchanged
slt $5, $26, $3
bne $5, $0, print
; We are lowercase, add $20
add $3, $3, $20

print: sw $3, 0($28)
       beq $0, $0, loop

end: j r $31
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.

Solution:

; $27 is stdin, $28 is stdout
lis $27
.word 0xffff0004
lis $28
.word 0xffff000c

; $24 is EOF
lis $24
.word -1

; $4 is 4
lis $4
.word 4

; $26 is the initial value of $30
add $26, $30, $0

; Load characters until EOF is encountered
loop: lw $3, 0($27)
beq $3, $24, end

; Push the character
sw $3, -4($30)
sub $30, $30, $4

; Repeat
beq $0, $0, loop

end:

; Pop characters until $30 is back where it started
loop2: beq $26, $30, end2

; Pop a character
add $30, $30, $4
lw $3, -4($30)
;Print the character
sw $3, 0($28)

;Repeat
beq $0, $0, loop2

end2:    jr $31
Problem 7 - Functions and recursion in MIPS

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

Solution:

;The first instance of fact is implicitly called by the start of the program

fact:  sw $31,  -4($30)
       sw $1,   -8($30)
       sw $11, -12($30)
       lis $31
       .word 12
       sub $30, $30, $31

       lis $11
       .word 1

       ;If $1 = 0, base case of $3 = 1
       bne $1, $0, recur
       add $3, $11, $0
       beq $0, $0, clean

       ;Call fact with $1 - 1
       recur: sub $1, $1, $11
              lis $31
              .word fact
              jalr $31

              ;Restore value of $1
              add $1, $1, $11

              ;Multiply previous answer by $1 to get new factorial
              mult $3, $1
              mflo $3

       clean: lis $31
              .word 12
              add $30, $30, $31

              lw $11, -12($30)
              lw $1,  -8($30)
              lw $31, -4($30)
              jr $31