MIPS Assignment 10

Recursion

Program

Write a program that lets the user input an integer, then outputs a series of outputs corresponding to the result of calling a recursive function multiple times.

The Input

As with previous assignments, inputs are entered on a single line.

The user will enter one integer. This integer will specify how many iterations of the recursive function are outputted (i.e., how many lines of output).

The Output

Your main loop should call your recursive function N times (where N is the user-inputted integer), printing the final result of each call on its own line. These calls should start at 0, such that the recursive function is called with 0, 1, 2, etc, … up to N-1. For example, if the user inputs a 6, your recursive function would be called with 0, 1, 2, 3, 4, and finally 5.

As with all previous assignments, each outputted number should be followed by a newline character ('\n' aka ASCII 10 aka Hex 0x0A), so all outputted numbers are on their own lines.

If, for example, the user wanted to display 6 outputs from your recursive function, they would enter a 6, and a correct program would output the following:

```
0
1
1
2
3
5
```

What's inside the Recursive Function?

You may be wondering what the function's algorithm is. It's quite simple:

1. Suppose for a moment: Our recursive function has the following prototype: ***f***(X)
2. If X is 0, return 0 (this is a base case)
3. If X is 1, return 1 (this is also a base case)
4. If X is more than 1, return the result of recursively calling: ***f***(X – 1) + ***f***(X – 2)

We then call ***f*** repeatedly from within our main loop, starting with 0, until we have called it the correct number of times (specified by user input).

To help clarify, here's a very simple python implementation:

```
#   Our looping function
#   This doesn’t need to be a function in MIPS; You can just loop
def do_loop(count):

    #     Remember: Python will produce i values in the range [0, i - 1]
    for i in range(count):
          print(f(i))


#   Our main recursive function
def f(x):
    if x == 0 or x == 1:
          return x

    return f(x - 1) + f(x - 2)


#   Get user input and call
inputted_count = input()
do_loop(inputted_count)
```

Special Instructions

The point of this assignment is to get you more comfortable using the stack to save variables when calling recursive functions. Your program should contain four main areas:

1. The beginning of your program, where the user enters an input.
2. A loop area that repeatedly calls our recursive function and prints its result the correct number of times.
3. Our recursive function that takes in one argument (X), and returns the result of its algorithm as stated above.
4. The end of the program that exits properly.

Your recursive function ***<u>will be graded by hand</u>***, above and beyond any penalties incurred during the automated grading phase. It will be graded on the following:

- Adherence to discussed register conventions when:
    - Providing arguments to your function
    - Preserving variables across function calls
    - Returning values from your function
    - Saving to the stack inside your function

- Utilization of the stack (i.e., At least one variable should be placed inside an $sx register, and depend on the stack to stay alive).
- Use of recursion in your function (i.e., don’t try to be clever and avoid recursion by looping).
- Online plagiarism checks.
- Any other Shenanigans that sidestep the stated purpose of this assignment.

Hints

Here are some hints that may or may not be useful:

- If your program takes an incredibly long time for a user input of 30, you’re probably doing it right.
- You’ll probably want to review Chapter 2 and its corresponding slideshow if you get stuck.

Assignment Tag

For this assignment, use the following *Assignment tag*: *Mips10Recursion*