COE0147 Spring 2014
Lab 5: Using Stacks and Recursive Functions

You should submit your own solution via email according to the instructions on your TA's website. You may collaborate with a partner, but each person must electronically turn in his/her own copy of the lab, with the name of your partner. The due date is mentioned on the website. As usual, you have to submit your answers/files in a single zipped archive using the lab submission website.

In this lab, you will first write a few (non-recursive) functions that use the stack. In the second part, you will have to write recursive functions. Functions must comply with MIPS calling conventions, such as:

- Pass parameters via the $a* registers.
- Put the return value in $v* registers.
- Save registers on the stack as necessary (i.e., if your function uses any $s* registers, it should save and restore $s* registers using a prologue/epilogue).
- Save $ra on the stack, if the function is a non-leaf function.

Part 1: Using Stacks

In this part, you will have to manipulate a string. Your program will be required to replace four characters in the input string with the character '*' (0x2A). The catch is that you have to randomly select the four characters. The input string will be at least 10 characters and at most 63 characters long (excluding the NULL byte). Additionally it will not contain any '*' when it is entered by the user. You can declare a buffer as follows to store the input string from the user:

```
.data
input_str: .space   64
```

After you have received the input string, you have to find the length of it. Say the length of the string is N. To randomly replace a character of this string with a '*' character, you have to generate a random number x in the range 0 to N-1. This x will be used as the index of the character in the string that you will be replacing next. You have to do these steps (i.e., generate a random number for indexing and replace the character with a '*' at that index) four times. A tricky point here is: after you generate a random index, you have to check if the character is not already a '*'. If the character is '*', then you have to skip replacing it and generate another random index. Make sure there are four random '*'s in your output. For the same input string, the output string can be different each time you run the program because of the randomness. The output format is shown in the sample output:

```
Enter your string?
University of Pittsburgh                ← input from user (no need to print)
Here is your output?
Un*versity*of Pi*tsbu*gh
```

Because of the random indexes, another output of the same program could be:

```
Enter your string?
University of Pittsburgh                    ← input from user (no need to print)
Here is your output?
Unive**ity of Pi*tsbur*h
```

For submission, name your solution file as **lab5part1.asm**. Please do not forget to put the **.asm** extension and do not use any upper-case letter in the filename.

Part 2: Recursive Functions (Fibonacci Sequence)

The Fibonacci sequence is a special sequence of numbers discovered by mathematician Leonardo Fibonacci. In the Fibonacci sequence, each number is the sum of the two previous numbers. Its function can be defined as following:

$$Fibonacci(n)=Fibonacci(n-1)+Fibonacci(n-2) \quad \text{if } n>2$$
$$Fibonacci(n)=1 \quad \text{if } n <= 2$$

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…..**
Fibonacci(n)	1	1	2	3	5	8	13	…..

The Fibonacci sequence function can expressed recursively, as shown in the following pseudo-codes:

```
int Fibonacci(int n) {
        if (n <= 2) {
                return 1;               //The base case stops the recursion
        } else {
                return Fibonacci(n-1)+Fibonacci(n-2);
        }
}
```

Base your solution on the pseudo-codes above, translate the Fibonacci function into MIPS assembly. **Your code must call itself recursively (TAs will explicitly check for 'jal' instructions inside the function).**

Your function must find Fibonacci(n-1) and Fibonacci(n-2). Call a function that adds the two results together and return the sum. The Fibonacci function will save/restore any $s registers that it defines.

Sample output:

```
Enter the sequence index?
8                                   ← input from user (no need to print)
The Fibonacci value is:
21
```

For submission, name your solution file as **lab5part2.asm.** Please do not forget to put the **.asm** extension and do not use any upper-case letter in the filename.

[1] http://en.wikipedia.org/wiki/Recursion_(computer_science)#Fibonacci