Assignment #4

(max = 100)

A long time ago, a friend of mine showed me a proof having to do with what he called "Polish" sequences. To generate a Polish sequence, you start with any positive integer. You square the individual digits of this number and add those squares, producing the next number in the sequence. This procedure repeats (theoretically) forever.

Here are some Polish sequences generated for a few positive integers:

308, 73, 58, 89, 145, 42, 20, 4, 16, 37, 58, …
5121, 31, 10, 1, 1, …
27, 53, 34, 25, 29, 85, 89, 145, 42, 20, 4, 16, 37, 58, 89, …
12345678, 204, 20, 4, 16, 37, 58, 89, 145, 42, 20, …

It can actually be proven that the Polish sequence for an arbitrary positive integer will always terminate in the infinite sequence

1, 1, 1, …

or in the following repeating sequence of eight integers

20, 4, 16, 37, 58, 89, 145, 42, 20, …

Of course, the repeating sequence would not have to begin with 20 … it could begin with any of the 8 numbers that appear in that sequence.

You are to write a program that will allow us to test this claim. Below, I am starting you out with a little driver program (this program also appears on the course website under Course Materials as Polish.asm). I want you to add two procedures: (1) one called "terms" (called by the main program) whose job is to display the first 16 terms in a sequence that starts with the value that it is passed (in $a0) and (2) a function called "Polish" that will be called by the procedure "terms" … this function will actually calculate and return (in $v0) the next term in a sequence, given that the current term is the value that is passed to it (in $a0). Remember that since you have procedures calling other procedures, you need to be concerned with preserving and restoring certain values on the stack.

Here is the little driver program:

```
# Your name/date
# Appropriate documentation

# insert your terms procedure and your Polish function here

# Driver program provided by Stephen P. Leach -- written 10/03/07
```

```
main:  la     $a0, intro    # print intro
       li     $v0, 4
       syscall

loop:  la     $a0, req      # request value of n
       li     $v0, 4
       syscall

       li     $v0, 5        # read value of n
       syscall

       ble    $v0, $0, out  # if n is not positive, exit

       move   $a0, $v0      # set parameter for terms procedure

       jal    terms         # call terms procedure

       j      loop          # branch back for next value of n

out:   la     $a0, adios    # display closing
       li     $v0, 4
       syscall

       li     $v0, 10       # exit from the program
       syscall

       .data
intro: .asciiz  "Welcome to the Polish sequence tester!"
req:   .asciiz  "\nEnter an integer (zero or negative to exit): "
adios: .asciiz  "Come back soon!\n"
```

Start with this code and add your procedures. You may want to add a few items in the data segment (to assist the terms procedure in producing the appropriate output), but don't change the portion of the code that I am providing. Don't forget to document your code! We haven't officially covered some of the needed arithmetic operations, but the following should be enough information to get you going (the last two are actually pseudoinstructions):

```
       mul    $t1, $t2, $t3 # $t1 = $t2 * $t3
       div    $t1, $t2, $t3 # $t1 = $t2 / $t3 (truncated)
       rem    $t1, $t2, $t3 # $t1 = $t2 % $t3 (remainder from $t2 / $t3)
```

Feel free to use other instructions and pseudoinstructions that appear in Appendix A. Submit the file called Polish.asm. (100 points)

Your program will be evaluated as follows:

Documentation (20 points)
Appropriate style and indentation (10 points)
Appropriate algorithm (20 points)
Proper results (30 points)
Proper use of stack when making function calls (20 points)

Here is a sample execution from my solution:

```
Welcome to the Polish sequence tester!
Enter an integer (zero or negative to exit): 308
First 16 terms: 308 73 58 89 145 42 20 4 16 37 58 89 145 42 20 4
Enter an integer (zero or negative to exit): 5121
First 16 terms: 5121 31 10 1 1 1 1 1 1 1 1 1 1 1 1 1
Enter an integer (zero or negative to exit): 27
First 16 terms: 27 53 34 25 29 85 89 145 42 20 4 16 37 58 89 145
Enter an integer (zero or negative to exit): 12345678
First 16 terms: 12345678 204 20 4 16 37 58 89 145 42 20 4 16 37 58 89
Enter an integer (zero or negative to exit): 0
Come back soon!
```