EE 2310 Homework #7 –Complex SPIM Programs

Name:_____ Student Number:_____

For the three problems given, develop MIPS programs that satisfy the specification given in the problem statement.

1. **Develop a looping program that will take a series of decimal numbers and arrange them in ascending order from smallest to largest. The sequence of nine numbers is shown below. Of course they could be input via the keyboard but for simplicity, you should simply put them in the data declaration. Use the space at the right for your program.**

Numbers: 5,3,8,9,2,4,1,7,6,3,8

When the numbers are ordered correctly in the string, you do not have to print them out – simply leave them in order in memory.

```
# Order Program
# Arranges a given set of numbers
# in ascending numerical order.

        .text
main:
```

```
        .data
```

2. **In class (Lecture 17) we saw one way to use a recursive function to calculate a factorial function (n!). Another way to do this is to define the factorial function as:**

n! = n*(n-1)!, where 0!=1.

Then n! can be calculated similarly to the way we calculated the function of x^y. That is, we use the procedure call jal to set up the number of loops necessary to calculate the n! result, and then use the jr ("procedure uncall") to do the actual math. Using the definitions listed above for n! and 0!, construct a program that uses recursion to calculate n! in the manner suggested above.

The number will be input from the console. As with the n! program shown in class, note that the number must be 12 or less, since we are not doing floating-point calculations.

```
# N! Program
# Calculates n! for any integer
# of 12 or less (due to use of
# only fixed-point calculations)

        .text
main:
                                    .data
input:  .asciiz "Input integer (0-12): "
ans:    .ascii " factorial is "
```

3. **In the space provided, write a program to set up an encryption function to encode "hello world" and print it to the console in encoded form. Presumably, some secret agent will later read and decode it, perhaps performing some vital action based on the content! Well, probably not. Anyway, that is your task. Use the following encryption key:**

   Control and punctuation ASCII codes are represented by numbers which are 256 times their actual numerical value plus 65536. That is, for example, space (0x20, = 32_{10}) would be calculated as:

   Space = 32*256+65536 = 73,728. Of course, this number would be stored in hex, but would be printed out in decimal.

   Small alphabetical symbols are represented as numbers which are 1024 times their value plus 4096. Thus a (0x61, = 97_{10}) would be:

   a = 97*1024+4096 = 103424.

   Assume no other symbols (such as large letters) are represented, so that you only have to worry about lower-case letters and non-lower-case letters.

   Your final output should be a string of numbers to the console which represent "hello world\n" in the coded form defined above. Output a "CR/LF" between each word.

```
# Encryption Program
# Encrypts "hello world\n"

        .text
main:
```

```
        .data
string: .asciiz "hello world\n"
code:   .word 0,0,0,0,0,0,0,0,0,0,0
crlf:   .asciiz "\n"
```