ECE 1381: Microcontrollers and Embedded Systems
Spring 2024 – Professor Joe Camp

HW4: Stacks and Tables
Due 4/11, 9:30am (submitted via Canvas)

1. (25 points) Convert the factorial program in Chapter 3 into a subroutine, using empty descending stacks. Pass arguments to the subroutine using the pass-by-stack technique. Please show your code works with the Red Pitaya tools by grabbing a screen shot with your name somewhere on the screen. NOTE: You can use the subroutine functionality in Chapter 3, you just need to write code around it to use empty descending stacks to pass inputs and outputs.

2. (25 points) Imagine the sine lookup table used in Example 12.1 of the book (the sine example used in lecture) had values from 0 through 180 degrees (instead of 0 to 90 degrees as discussed). Reduce the code from example 12.1 to work for this new table (not check in quadrants but rather check for a value from 0 to 180 or from 180 to 360 and compensate accordingly). Show your code works with your Red Pitaya by grabbing a screen shot with your name somewhere on the screen.

3. (25 points) A binary search only works if the values in the list are sorted. A bubble sort is a simple way to sort entries. The basic idea is to compare two adjacent entries in a list—call them entry[j] and entry[j+1]. If entry[j] is larger, then swap the entries. If this is repeated until the last two entries are compared, the largest element in the list will now be last. The smallest entry will ultimately get swapped, or "bubbled" up to the top. The algorithm could be described in C as:

```
last = num;
while (last > 0) {
        pairs = last – 1;
        for (j = 0; j <= pairs; j++) {
                if (entry[j] > entry [j+1]
                {
                        temp = entry[j];
                        entry[j] = entry[j+1];
                        entry[j+1] = temp;
                        last = j;
                }
        }
}
```

   Here, num is the number of entries in the list. Write an assembly language program to implement a bubble sort algorithm, and test it using a list of 10 elements. Each element should be a word in length. Show your code works with your Red Pitaya by grabbing a screen shot with your name somewhere on the screen.

4. (25 points) Using the bubble sort algorithm written in the previous problem, write an assembly program that sorts entries in a list and then uses a binary search to find a particular key. Remember that your sorting routine must sort both the key and the data associated with each entry. Create a list with at least 10 entries, and data for each key should be at least 16 bytes of information. Show your code works with your Red Pitaya by grabbing a screen shot with your name somewhere on the screen.