8. [**10 pts**] The `fibolist()` function, shown here, calculates elements of the Fibonacci series recursively and *also* fills in an array with those elements as it goes. Write `fibolist()` in ARMv8 (or LEGv8) assembly language. Follow the ARM calling conventions for arguments, returned value, saved registers, return address, and other registers, as listed on the LEGv8 reference data card.

   Write efficient code, not just the output of the gcc compiler's "-S" flag.

```
long int fibolist(long int *lst, int n)
{
    if (n < 2) {
        return 1;
    } else {
        lst[n] = fibolist(lst, n-1)
                 + fibolist(lst, n-2);
        return lst[n];
    }
}
```

9. [**10 pts**] Translate the `leaf()` function shown here, into ARMv8 assembly language.

   Follow the ARMv8 calling conventions for arguments, returned value, saved registers, return address, and other registers, as listed on the ARMv8 reference data card.

```
long int leaf(long int *lst, int len)
{
  long int d;
  d = lst[0] - lst[ len-1 ];
  return d;
}
```

10. [**10 pts**] Translate ("disassemble") this ARMv8 assembly routine into the corresponding C function:

```
// Expects:
//     x0: long int *array
//     x1: long int length
// Returns:
//     x0: long int answer
.equ FrameSize, 16
    .text
    .global looper
looper:
    stp  x29, x30, [sp, -FrameSize] !    // set up frame
    mov  x29, sp                         // set fp

    ldr  x9, [x0]        // start w/ 1st value
    mov  x2, #1          // loop counter
    b    check
each:
    ldr  x3, [x0, x2, lsl #3]    // get each array element
    cmp  x3, x9
    b.ge next
    mov  x9, x3
next:
    add  x2, x2, #1      // increment loop counter
check:
    cmp  x2, x1
    b.lo each
done:
    mov  x0, x9
    mov  sp, x29                         // restore sp
    ldp  x29, x30, [sp], FrameSize       // tear down frame
    ret
#--------
```

Here is a basic program that uses the functions given in the previous questions, and a sample run showing its (and their) output:

tester.c

```c
/*
* Simple main() that exercises coding problems from CS330 exam 1.
* 2020-02-26
*/
#include<stdio.h>
#include<stdlib.h>   // atoi()

long int fibolist(long int *lst, int n);
long int leaf(long int *lst, int len);
long int looper(long int *array, long int length);

int main(int argc, char **argv)
{
    int len = (argc > 1) ? atoi(argv[1]) : 5;
    printf("array length: %d\n", len);

    // Demo question 7:
    long int fiboseq[len];
    // starting values for Fibonacci series:
    fiboseq[0] = 1;
    fiboseq[1] = 1;

    // Demo question 8:
    for (int i = 0; i < len; i++) {
        long int fibo;
        fibo = fibolist(fiboseq, i);
        printf("fibolist(fiboseq, %d): %ld\n", i, fibo);
    }
    for (int i = 0; i < len; i++)
        printf("fiboseq[%d]: %ld\n", i, fiboseq[i]);

    // Demo question 9:
    long int looperans = looper(fiboseq, (long int)len);
    printf("looper(myArray, %d): %ld\n", len, looperans);

    return 0;
}
```

sample runs

```
$ as  -o looper.o looper.s
$ gcc -Wall -o tester tester.c fibolist.c leaf.c   looper.o
$  ./tester 6
array length: 6
fibolist(fiboseq, 0): 1
fibolist(fiboseq, 1): 1
fibolist(fiboseq, 2): 2
fibolist(fiboseq, 3): 3
fibolist(fiboseq, 4): 5
fibolist(fiboseq, 5): 8
fiboseq[0]: 1
fiboseq[1]: 1
fiboseq[2]: 2
fiboseq[3]: 3
fiboseq[4]: 5
fiboseq[5]: 8
looper(myArray, 6): 8
$  ./tester
array length: 5
fibolist(fiboseq, 0): 1
fibolist(fiboseq, 1): 1
fibolist(fiboseq, 2): 2
fibolist(fiboseq, 3): 3
fibolist(fiboseq, 4): 5
fiboseq[0]: 1
fiboseq[1]: 1
fiboseq[2]: 2
fiboseq[3]: 3
fiboseq[4]: 5
looper(myArray, 5): 5
$
```