CS 164, Fall 2020 **CS 164: Homework #7** **P. N. Hilfinger**

Due: Fri, 20 November 2020

Put answers to this problem in files `hw7-1.txt`, `hw7-2.txt`, and `hw7-3.txt`.

1. I produced the following program using `gcc -S foo.c` for the RISC-V architecture:

```
f:
        addi    sp,sp,-48
        sw      s0,44(sp)
        addi    s0,sp,48
        sw      a0,-36(s0)
        sw      a1,-40(s0)
        sw      zero,-20(s0)
        sw      zero,-24(s0)
        j       .L2
.L3:
        lw      a5,-24(s0)
        slli    a5,a5,2
        lw      a4,-36(s0)
        add     a5,a4,a5
        lw      a5,0(a5)
        lw      a4,-20(s0)
        add     a5,a4,a5
        sw      a5,-20(s0)
        lw      a5,-24(s0)
        addi    a5,a5,1
        sw      a5,-24(s0)
.L2:
        lw      a4,-24(s0)
        lw      a5,-40(s0)
        blt     a4,a5,.L3
        lw      a5,-20(s0)
        mv      a0,a5
        lw      s0,44(sp)
        addi    sp,sp,48
        jr      ra
```

Produce a plausible definition (in C) of function `f`, one that might have produced this output. The function does return a value. (You may display the result in Python if you wish, using Python lists for arrays.)

2. In lecture, we talked about *array descriptors,* which are data structures containing all the information one needs to access (get the address of) an array element `A[i,j]` in an implementation that allocates all elements of a new array contiguously. In C, multidimensional arrays are composed of rows of rows, so that `A[i,j]` (or `A[i][j]` in C) is located at address$(A_{0,0}) + N \cdot S \cdot i + S \cdot j$, where the array in `A` is $M \times N$ and each element has size S. Thus, the three constants data address$(A_{0,0})$ (the virtual origin), $N \cdot S$ (the row stride), and S (the column stride) can be precomputed into an *array descriptor,* which the program can use to generate array accesses and can pass as a parameter to functions that expect to receive the array as a by-reference parameter. Show the RISC V code that you'd use to access array element `A[i][j]`, assuming that d, t_i, and t_j are registers containing the address of the array descriptor for `A`, the value of `i`, and the value of `j`, respectively.

3. By constructing appropriate array descriptors, one can give different views of an array. Describe how to compute the constructors to create the following views (we don't need the actual code, just the calculations it must do).

a. Suppose that a certain array descriptor contains the information (VO, S_1, S_2) for accessing two-dimensional array `B`. Show how to create a new array descriptor that accesses column number j of `B`. This will be a one-dimensional array descriptor (having only one stride).

b. Show how to create a new array descriptor that accesses the transpose of `B`.

c. Show how to create a new array descriptor (for array view `B'`) that accesses the rows and columns of `B` in reverse, so that `B'[0,0]` is the same as the last column of the last row of `B`.