1 a Consider the following spatial program for kernel `madd`:

```
SCSVar a=io.input(``a'', SCSFix(32,16));
SCSVar b=io.input(``b'', SCSFix(32,16));
SCSVar fb = newFeedback(SCSFix(32,16));
SCSVar r = fb * a + b;
io.output(``r'', r, SCSFix(32,16));
```

Assume the `newFeedback` call creates a feedback loop for a multiply-add computation. Write the pseudo CPU code to make this kernel compute a polynomial: $\sum c_i x^i$. Next, please show the minimal modification to the kernel above and the CPU pseudo code to make the computation about twice as efficient.

b Assume a program with three input variables with dynamic ranges of: $a = [-20, 1000], b = [-500, 500], c = [1, 10]$. Write down the OpenSPL definition for both floating point SCSFloat(total bits, mantissa bits) and fixed point SCSFix(total bits, fractional bits) for these variables with a minimum number of bits, to maintain a precision to within an absolute error of 2^{-10}. Explain the term "sensitivity" of outputs to changes in inputs.

The two parts carry equal marks.

2a A two-dimensional set of five coefficients `c[]` is called a star stencil. Such a star stencil can be applied to an array of 1,000 by 1,000 single precision floating point numbers, in a single pass, called a timestep. Ignoring the boundaries, we have:

```
for i=1 to 998
 for j=1 to 998
     anext[i,j]=a[i,j-1]*c[0]+a[i-1,j]*c[1]+
       a[i,j]*c[2]+a[i+1,j]*c[3]+a[i,j+1]*c[4]
```

How large an FMEM buffer is needed to store a sufficiently large window for such a pass? Show spatial computing pseudo code using fixed point with 32 bits with 10 fractional bits and a minimal amount of FMEM, to implement such a pass. Use a `FIFO(a,d)` function to delay an SCSVar `a` by `d` cycles, and a `DELAY(a)` to delay by one cycle.

b Explain how you would parallelize the implementation from part 1 into four kernels running all in parallel. How would the implementation change? How would you optimize the implementation to take as little space as possible?

c Comparison in Space:

i) Given bidirectional LMEM bandwidth of 10GB per second and a spatial computing substrate operating at 500MHz, how long would it take to calculate 10,000 timesteps for the parallel implementation above?

ii) What is the optimal number of stencils running in parallel to saturate the 10GB per second bi-directional bandwidth? Assume that a single unit in a datacenter (called 1U) holds eight dataflow engines, each with the optimal number of stencils to saturate the bi-directional bandwidth, running in parallel, and assume that a 1U CPU node with 24 cores running at 2GHz maintains an effective aggregate 9 GFLOPS at a similar power consumption as the dataflow engine node. What is the speed and power efficiency comparing a 1U dataflow node to a 1U CPU node?

The three parts carry, respectively, 30%, 30%, and 40% of the marks.

3 a In this question we compare a modern microprocessor with a dataflow engine in terms of a new metric called BCAP. BCAP is the product of bandwidth and capacity of a memory. In the case of multiple memories, BCAP is the sum of the individual BCAPs of the memories.

i) Let us assume a dataflow engine with 3,000 blocks of FMEM memory, each holding 20Kbits and each with two 40bit ports, allowing simultaneous accesses at 600MHz. FMEM is fast, returning a result on the cycle following the receipt of an address. What is the aggregate FMEM capacity and bi-directional FMEM bandwidth for a 1U dataflow appliance with eight dataflow engines? What is the BCAP of a 1U dataflow appliance in $[Bytes^2 persecond]$?

ii) Now for the CPU side, calculate the BCAP of a Haswell CPU server with

* 24 cores per 1U server, with each core driving a private L1 and L2 cache. The L1 cache has 128KB and 700GB/s bandwidth, and the L2 has 1MB with 200GB/s

* The server has two CPU chips, each with an L3 cache with 6MB and 100GB/s access bandwidth.

What is the BCAP difference between the dataflow engine and the CPU system per 1U enclosure?

b i) We would like to implement a classification application which for a stream of 13-bit input values `in` checks their belonging to classes via a table lookup. Assume that the FMEM blocks are predefined and can be accessed via an `FMEM(mem, port, address)` function call, where `mem` is the number of the FMEM memory. The result of the table lookup is a simple yes/no decision bit, stored in a SCSVar vector `vec`. Write down the pseudocode for the instantiation of the FMEM blocks.

ii) How many classes can be checked in realtime, absorbing one input on every cycle? What is the effectively realised memory access bandwidth for this application?

The two parts carry equal marks.