Name: ____________________________

1. This problem concerns the CUDA code for Gaussian elimination, Sec. 11.5.2. Assume that the code that calls the kernel will have quantities A, b and n at the beginning of Sec. 11.5 stored in the variables **a**, **b** and **n**, respectively. The array **a** is one-dimensional, length n^2. We have another array **ab**, one-dimensional, length $n(n+1)$, corresponding to the argument of the same name in the kernel.

(a) (20) Fill in the blank in the following statement:

```
dim3 dimBlock( blank ,1,1);
```

(b) (20) The code preparing **ab** will include the following, in which you will fill in the blank:

```
for (j = 0; j < n; j++) ab[ blank ] = b[j];
```

2. (20) Consider applying the smoothing idea, Sec. 13.5.1, to audio, in the time domain. We could adapt the code in Sec. 4.14 for this. The argument **k** now will be the number of neighbors to smooth with, using **k/2** data points before and after the given point. We would delete much of the code. In particular, replace line 48 by

```
perlen = k;
```

deleting line 62.

We would have to add a crucial line. State the line number after which the new statement would be added, and state what single line should be added; don't worry about "corner cases," say what happens near the ends of the array.

3. (40) Use "Snow " (the portion of the R library **parallel** that was derived from the old **snow** library) to implement the run-length coding decompression algorithm in Sections 10.5 and 10.6. The "declaration" of your function will be

```
decomp <- function(x,cls)
```

where **x** is the compressed vector, and **cls** is a Snow cluster.

Your code need not be optimal, just parallel and correct. Submit just the function itself in the end, but you may wish to temporarily put in a test case so you can try your code through OMSI.

Solutions:

1.a n

1.b (j+1) * (n+1)

2. After line 56, insert

```
x[perstart + k/2] = xbar;
```

3. Outline:

```
split x into chunks for the workers
each worker does a straightforward decompression of its chunk
apply Reduc(c, ) to what is returned from the workers
```