Name: ____________________________

Directions: **Work only on this sheet** (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing.

Unless otherwise stated, give numerical answers as expressions, e.g. $\frac{2}{3} \times 6 - 1.8$. Do NOT use calculators.

1. (35) The code below does an in-place transpose of a square matrix. (Note: *No unnecessary computation is done.*) Fill in the blanks.

```
#include <stdio.h>
#include <stdlib.h>
#include <cuda.h>

__global__ void transpairs(int *m, int n, int nth)
{
   int thn = blockIdx.x;  // thread number
   // this thread will handle one below-diagonal element and
   // its "mate" above the diagonal;
   // first, determine the row and column of
   // the below-diagonal one
   int i,j,count=-1, done = 0;
   for (i=0; i < n-1;i++) {
      for (j=0; j<=i; j++) {
         count++;
         if (count == thn) {
            done = 1;
            break;
         }
      }
      if (done) break;
   }
   i++;
   -------------------------------------------------
   int tmp = m[w1];
   m[w1] = m[w2];
   m[w2] = tmp;
}

int main(int argc, char **argv)
{
   int n = atoi(argv[1]);  // number of matrix rows/cols
   int *hm, *dm;
   int msize = n * n * sizeof(int);
   hm = (int *) malloc(msize);
   // as a test, fill matrix with consecutive integers
   int t = 1,i,j;
   for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++) {
         hm[i*n+j] = t++;
      }
   }
   cudaMalloc((void **)&dm,msize);
   cudaMemcpy(dm,hm,msize,cudaMemcpyHostToDevice);
   -------------------------------------------------;
   dim3 ----------------------------------------------------------;
   dim3 ----------------------------------------------------------;
   transpairs<<<dimGrid,dimBlock>>>(dm,n,nth);
   cudaThreadSynchronize();
   cudaMemcpy(hm,dm,msize,cudaMemcpyDeviceToHost);
   if (n < 10)
      for(int i=0; i<n; i++)
         for (int j = 0; j<n; j++) printf("%d\n",hm[n*i+j]);
   free(hm);
   cudaFree(dm);
}
```

2. Consider the Edgar matrix multiplication routine, with

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix} \qquad (1)$$

and

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \end{pmatrix} \qquad (2)$$

Further suppose that **BLOCK_SIZE** is 2. Take row and column numbers to start at 0, e.g. the (1,0) element of B is 5. Consider the calculation of the (1,1) element of the product C.

(a) (20) Give the "coordinates" of the thread handling this computation, i.e. the values of variables **bx**, **tx** etc. in the code.

(b) (20) During this computation, **Csub** will take on various values. List the first one that occurs after 0.

3. (25) There is an error concerning the call to **__syncthreads()** in the CUDA prime-finding program, causing an inefficiency though not incorrect results. State what it is.

Solutions:

1.

```
int w1 = i*n+j, w2 = j*n+i;
...
int nth = n*(n-1)/2;
dim3 dimGrid(nth,1);
dim3 dimBlock(1,1,1);
```

2a. Thread (1,1) within block (0,0).

2b. The entire computation is a sum of six products, taken two at a time. With the first two, the terms will be 7 x 2 and 8 x 6, making **Csub** 62.

3. The call should be moved inside the **for** loop, so that the check **sprimes[m] != 0** is valid.