1. Describe a cache-efficient algorithm for computing the matrix product

$$C^{n\times n} = A^{n\times n}.B^{n\times n}$$

for an external memory model with parameters M, B.
Note: Use the normal matrix multiplication and not the fancy sub-cubic Strassen kind of algorithm.

Solution: For the case $M \geq B^2$, tile the $N \times N$ matrix using $\sqrt{M} \times \sqrt{M}$. Then the matrix multiplication can be re-written as

$$C^{\frac{N}{\sqrt{M}},\frac{N}{\sqrt{M}}} = A^{\frac{N}{\sqrt{M}},\frac{N}{\sqrt{M}}} \otimes B^{\frac{N}{\sqrt{M}},\frac{N}{\sqrt{M}}}$$

where the matrix multiplication $\otimes$ involves sub-matrices of size $\sqrt{M}\times\sqrt{M}$. There will be two loops, the outside one for the $\frac{N}{\sqrt{M}} \times \frac{N}{\sqrt{M}}$ tiles. And the inner loop for the multiplication of $\sqrt{M}\times\sqrt{M}$ tiles. The computation corresponding to the inner loop can be performed inside the internal memory. So the overall number of I/O's will be the cost of reading and writing the $(\frac{N}{\sqrt{M}})^3$ tiles corresponding to the multiplication and addition. Note that each tile is accessed $\frac{N}{\sqrt{M}}$ times and there are $(\frac{N}{\sqrt{M}})^2$ tiles. Moreover we need $\frac{M}{B}$ I/O's to access a tile. This leads to $(\frac{N}{\sqrt{M}})^3 \times \frac{M}{B} = O(\frac{N^3}{B\sqrt{M}})$ I/O's.

For the case, $B^2 \geq M \geq B$, we will create rectangular tiles of size $B \times \frac{M}{B}$ and the number of tile products (involving $\frac{M}{B} \times B$ and $B \times \frac{M}{B}$) can be shown as $\frac{N}{\frac{M}{B}} \times \frac{N}{B}\frac{N}{\frac{M}{B}}$ where each tile needs $\frac{M}{B}$ I/O's. So overall its $\frac{N^3}{M}$ I/O's. Therefore for $B^2 \geq M \geq B$, this is between $\frac{N^3}{B^2}$ to $\frac{N^3}{B}$.

2. Design a cache-oblivious algorithm for multiplying an $N \times N$ matrix by an N vector.

Solution: If we write the normal matrix vector multiplication as computing N inner products and an LRU policy then there will be $N \times (\frac{N}{B} + \frac{N}{B}) \leq O(\frac{N^2}{B})$ I/O's.
We do not need B in the program since LRU will force the block to be resident for the B multiplications and additions. We need an additional variable for the partial sums. A recursive solution will also yield similar result with the right base case.

3. Given a complete $k-ary$ tree (every internal node has k children), determine the pebbling complexity with m red pebbles. Note that an internal node, including the root, can be pebbled only if all its children have red pebbles. The leaf nodes are the input nodes and the root finally contains the output.

Solution: Assume $m \geq k$, else we won't be able to pebble. Following the example of FFT graph, one can show that in this case for any pebble p,

$$charge(p) \leq \frac{\log m}{\log k} \text{for a } k-ary \text{ complete tree } \implies \sum_p charge(p) \leq \frac{m \log m}{\log k} \text{for input of blocks of size } k$$

So the pebble complexity for the block m transfer

$$\geq (\frac{n}{k})\frac{\log k}{m \log m} + \frac{n}{m} \text{ (for the leaf nodes)}$$

or

$$\geq (\frac{n}{k})\frac{\log k}{\log m} + n \text{ (for block of size 1)}$$